Robot Localization and Map Building

Robot Localization and Map Building

Edited by
Hanafiah Yussof

In-Tech
intechweb.org

Published by In-Teh

In-Teh
Olajnica 19/2, 32000 Vukovar, Croatia

First published March 2010
Printed in India

Technical Editor: Sonja Mujacic
Cover designed by Dino Smrekar

Robot Localization and Map Building,
Edited by Hanafiah Yussof

p. cm.
ISBN 978-953-7619-83-1

Preface

Navigation of mobile platform is a broad topic, covering a large spectrum of different technologies and applications. As one of the important technology highlighting the 21st century, autonomous navigation technology is currently used in broader spectra, ranging from basic mobile platform operating in land such as wheeled robots, legged robots, automated guided vehicles (AGV) and unmanned ground vehicle (UGV), to new application in underwater and airborne such as underwater robots, autonomous underwater vehicles (AUV), unmanned maritime vehicle (UMV), flying robots and unmanned aerial vehicle (UAV).

Localization and mapping are the essence of successful navigation in mobile platform technology. Localization is a fundamental task in order to achieve high levels of autonomy in robot navigation and robustness in vehicle positioning. Robot localization and mapping is commonly related to cartography, combining science, technique and computation to build a trajectory map that reality can be modelled in ways that communicate spatial information effectively. The goal is for an autonomous robot to be able to construct (or use) a map or floor plan and to localize itself in it. This technology enables robot platform to analyze its motion and build some kind of map so that the robot locomotion is traceable for humans and to ease future motion trajectory generation in the robot control system. At present, we have robust methods for self-localization and mapping within environments that are static, structured, and of limited size. Localization and mapping within unstructured, dynamic, or large-scale environments remain largely an open research problem.

Localization and mapping in outdoor and indoor environments are challenging tasks in autonomous navigation technology. The famous Global Positioning System (GPS) based on satellite technology may be the best choice for localization and mapping at outdoor environment. Since this technology is not applicable for indoor environment, the problem of indoor navigation is rather complex. Nevertheless, the introduction of Simultaneous Localization and Mapping (SLAM) technique has become the key enabling technology for mobile robot navigation at indoor environment. SLAM addresses the problem of acquiring a spatial map of a mobile robot environment while simultaneously localizing the robot relative to this model. The solution method for SLAM problem, which are mainly introduced in this book, is consists of three basic SLAM methods. The first is known as extended Kalman filters (EKF) SLAM. The second is using sparse nonlinear optimization methods that based on graphical representation. The final method is using nonparametric statistical filtering techniques known as particle filters. Nowadays, the application of SLAM has been expended to outdoor environment, for use in outdoor's robots and autonomous vehicles and aircrafts. Several interesting works related to this issue are presented in this book. The recent rapid progress in sensors and fusion technology has also benefits the mobile platforms performing navigation in term of improving environment recognition quality and mapping accuracy. As one of important element in robot localization and map building, this book presents interesting reports related to sensing fusion and network for optimizing environment recognition in autonomous navigation.

This book describes comprehensive introduction, theories and applications related to localization, positioning and map building in mobile robot and autonomous vehicle platforms. It is organized in twenty seven chapters. Each chapter is rich with different degrees of details and approaches, supported by unique and actual resources that make it possible for readers to explore and learn the up to date knowledge in robot navigation technology. Understanding the theory and principles described in this book requires a multidisciplinary background of robotics, nonlinear system, sensor network, network engineering, computer science, physics, etc.

The book at first explores SLAM problems through extended Kalman filters, sparse nonlinear graphical representation and particle filters methods. Next, fundamental theory of motion planning and map building are presented to provide useful platform for applying SLAM methods in real mobile systems. It is then followed by the application of high-end sensor network and fusion technology that gives useful inputs for realizing autonomous navigation in both indoor and outdoor environments. Finally, some interesting results of map building and tracking can be found in 2D, 2.5D and 3D models. The actual motion of robots and vehicles when the proposed localization and positioning methods are deployed to the system can also be observed together with tracking maps and trajectory analysis. Since SLAM techniques mostly deal with static environments, this book provides good reference for future understanding the interaction of moving and non-moving objects in SLAM that still remain as open research issue in autonomous navigation technology.

Hanafiah Yussof

Nagoya University, Japan

Universiti Teknologi MARA, Malaysia

Contents

Visual Localisation of quadruped walking robots

Renato Samperio and Huosheng Hu
*School of Computer Science and Electronic Engineering, University of Essex
United Kingdom*

1. Introduction

Recently, several solutions to the robot localisation problem have been proposed in the scientific community. In this chapter we present a localisation of a visual guided quadruped walking robot in a dynamic environment. We investigate the quality of robot localisation and landmark detection, in which robots perform the RoboCup competition (Kitano et al., 1997). The first part presents an algorithm to determine any entity of interest in a global reference frame, where the robot needs to locate landmarks within its surroundings. In the second part, a fast and hybrid localisation method is deployed to explore the characteristics of the proposed localisation method such as processing time, convergence and accuracy.
In general, visual localisation of legged robots can be achieved by using artificial and natural landmarks. The landmark modelling problem has been already investigated by using predefined landmark matching and real-time landmark learning strategies as in (Samperio & Hu, 2010). Also, by following the pre-attentive and attentive stages of previously presented work of (Quoc et al., 2004), we propose a landmark model for describing the environment with "interesting" features as in (Luke et al., 2005), and to measure the quality of landmark description and selection over time as shown in (Watman et al., 2004). Specifically, we implement visual detection and matching phases of a pre-defined landmark model as in (Hayet et al., 2002) and (Sung et al., 1999), and for real-time recognised landmarks in the frequency domain (Maosen et al., 2005) where they are addressed by a similarity evaluation process presented in (Yoon & Kweon, 2001). Furthermore, we have evaluated the performance of proposed localisation methods, Fuzzy-Markov (FM), Extended Kalman Filters (EKF) and an combined solution of Fuzzy-Markov-Kalman (FM-EKF),as in (Samperio et al., 2007)(Hatice et al., 2006).
The proposed hybrid method integrates a probabilistic multi-hypothesis and grid-based maps with EKF-based techniques. As it is presented in (Kristensen & Jensfelt, 2003) and (Gutmann et al., 1998), some methodologies require an extensive computation but offer a reliable positioning system. By cooperating a Markov-based method into the localisation process (Gutmann, 2002), EKF positioning can converge faster with an inaccurate grid observation. Also. Markov-based techniques and grid-based maps (Fox et al., 1998) are classic approaches to robot localisation but their computational cost is huge when the grid size in a map is small (Duckett & Nehmzow, 2000) and (Jensfelt et al., 2000) for a high resolution solution. Even the problem has been partially solved by the Monte Carlo (MCL) technique (Fox et al., 1999), (Thrun et al., 2000) and (Thrun et al., 2001), it still has difficulties handling environmental changes (Tanaka et al., 2004). In general, EKF maintains a continuous estimation of robot position, but can not manage multi-hypothesis estimations as in (Baltzakis & Trahanias, 2002).

Moreover, traditional EKF localisation techniques are computationally efficient, but they may also fail with quadruped walking robots present poor odometry, in situations such as leg slippage and loss of balance. Furthermore, we propose a hybrid localisation method to eliminate inconsistencies and fuse inaccurate odometry data with costless visual data. The proposed FM-EKF localisation algorithm makes use of a fuzzy cell to speed up convergence and to maintain an efficient localisation. Subsequently, the performance of the proposed method was tested in three experimental comparisons: (i) simple movements along the pitch, (ii) localising and playing combined behaviours and c) kidnapping the robot.
The rest of the chapter is organised as follows. Following the brief introduction of Section 1, Section 2 describes the proposed observer module as an updating process of a Bayesian localisation method. Also, robot motion and measurement models are presented in this section for real-time landmark detection. Section 3 investigates the proposed localisation methods. Section 4 presents the system architecture. Some experimental results on landmark modelling and localisation are presented in Section 5 to show the feasibility and performance of the proposed localisation methods. Finally, a brief conclusion is given in Section 6.

2. Observer design

This section describes a robot observer model for processing motion and measurement phases. These phases, also known as *Predict* and *Update*, involve a state estimation in a time sequence for robot localisation. Additionally, at each phase the state is updated by sensing information and modelling noise for each projected state.

2.1 Motion Model

The state-space process requires a state vector as processing and positioning units in an observer design problem. The state vector contains three variables for robot localisation, i.e., 2D position (x, y) and orientation (θ). Additionally, the prediction phase incorporates noise from robot odometry, as it is presented below:

$$\begin{pmatrix} x_t^- \\ y_t^- \\ \theta_t^- \end{pmatrix} = \begin{pmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{pmatrix} + \begin{pmatrix} (u_t^{lin} + w_t^{lin})cos\theta_{t-1} - (u_t^{lat} + w_t^{lat})sin\theta_{t-1} \\ (u_t^{lin} + w_t^{lin})sin\theta_{t-1} + (u_t^{lat} + w_t^{lat})cos\theta_{t-1} \\ u_t^{rot} + w_t^{rot} \end{pmatrix} \tag{4.9}$$

where u^{lat}, u^{lin} and u^{rot} are the lateral, linear and rotational components of odometry, and w^{lat}, w^{lin} and w^{rot} are the lateral, linear and rotational components in errors of odometry. Also, $t-1$ refers to the time of the previous time step and t to the time of the current step.
In general, state estimation is a weighted combination of noisy states using both priori and posterior estimations. Likewise, assuming that v is the measurement noise and w is the process noise, the expected value of the measurement R and process noise Q covariance matrixes are expressed separately as in the following equations:

$$R = E[vv^t] \tag{4.10}$$

$$Q = E[ww^t] \tag{4.11}$$

The measurement noise in matrix R represents sensor errors and the Q matrix is also a confidence indicator for current prediction which increases or decreases state uncertainty. An odometry motion model, u_{t-1} is adopted as shown in Figure 1. Moreover, Table 1 describes all variables for three dimensional *(linear, lateral and rotational)* odometry information where $(\widehat{x}, \widehat{y})$ is the estimated values and (x, y) the actual states.

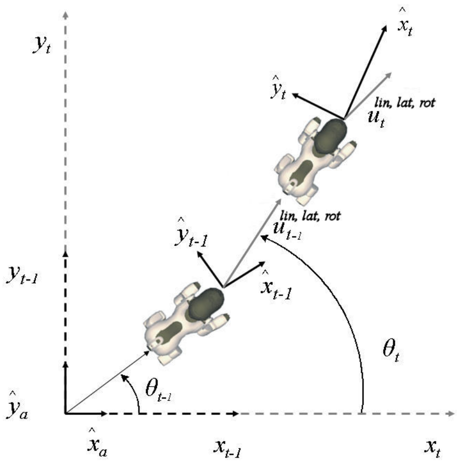


Fig. 1. The proposed motion model for Aibo walking robot

According to the empirical experimental data, the odometry system presents a deviation of 30% on average as shown in Equation (4.12). Therefore, by applying a transformation matrix W_t from Equation (4.13), noise can be addressed as robot uncertainty where θ points the robot heading.

$$Q_t = \begin{pmatrix} (0.3u_t^{lin})^2 & 0 & 0 \\ 0 & (0.3u_t^{lat})^2 & 0 \\ 0 & 0 & (0.3u_t^{rot} + \frac{\sqrt{(u_t^{lin})^2+(u_t^{lat})^2}}{500})^2 \end{pmatrix} \tag{4.12}$$

$$W_t = fw = \begin{pmatrix} cos\theta_{t-1} & -sen\theta_{t-1} & 0 \\ sen\theta_{t-1} & cos\theta_{t-1} & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.13}$$

2.2 Measurement Model

In order to relate the robot to its surroundings, we make use of a landmark representation. The landmarks in the robot environment require notational representation of a measured vector f_t^i for each i-th feature as it is described in the following equation:

$$f(z_t) = \{f_t^1, f_t^2, ...\} = \{ \begin{pmatrix} r_t^1 \\ b_t^1 \\ s_t^1 \end{pmatrix}, \begin{pmatrix} r_t^2 \\ b_t^2 \\ s_t^2 \end{pmatrix}, ...\} \tag{4.14}$$

where landmarks are detected by an onboard active camera in terms of range r_t^i, bearing b_t^i and a signature s_t^i for identifying each landmark. A landmark measurement model is defined by a feature-based map m, which consists of a list of signatures and coordinate locations as follows:

$$m = \{m_1, m_2, ...\} = \{(m_{1,x}, m_{1,y}), (m_{2,x}, m_{2,y}), ...\} \tag{4.15}$$

Variable	Description
$\widehat{x}_a$	x axis of world coordinate system
$\widehat{y}_a$	y axis of world coordinate system
x_{t-1}	previous robot x position in world coordinate system
y_{t-1}	previous robot y position in world coordinate system
θ_{t-1}	previous robot heading in world coordinate system
$\widehat{x}_{t-1}$	previous state x axis in robot coordinate system
$\widehat{y}_{t-1}$	previous state y axis in robot coordinate system
$u_t^{lin,lat}$	lineal and lateral odometry displacement in robot coordinate system
u_t^{rot}	rotational odometry displacement in robot coordinate system
x_t	current robot x position in world coordinate system
y_t	current robot y position in world coordinate system
θ_t	current robot heading in world coordinate system
$\widehat{x}_t$	current state x axis in robot coordinate system
$\widehat{y}_t$	current state y axis in robot coordinate system

Table 1. Description of variables for obtaining linear, lateral and rotational odometry information.

where the i-th feature at time t corresponds to the j-th landmark detected by a robot whose pose is $x_t = \begin{pmatrix} x & y & \theta \end{pmatrix}^T$ the implemented model is:

$$\begin{pmatrix} r_t^i(x,y,\theta) \\ b_t^i(x,y,\theta) \\ s_t^i(x,y,\theta) \end{pmatrix} = \begin{pmatrix} \sqrt{(m_{j,x}-x)^2 + (m_{j,y}-y)^2} \\ atan2(m_{j,y}-y, m_{j,x}-x)) - \theta \\ s_j \end{pmatrix} \tag{4.16}$$

The proposed landmark model requires an already known environment with defined landmarks and constantly observed visual features. Therefore, robot perception uses mainly defined landmarks if they are qualified as reliable landmarks.

2.2.1 Defined Landmark Recognition

The landmarks are coloured beacons located in a fixed position and are recognised by image operators. Figure 2 presents the quality of the visual detection by a comparison of distance errors in the observations of beacons and goals. As can be seen, the beacons are better recognised than goals when they are far away from the robot. Any visible landmark in a range from 2m to 3m has a comparatively less error than a near object. Figure 2.b shows the angle errors for beacons and goals respectively, where angle errors of beacons are bigger than the ones for goals. The beacon errors slightly reduces when object becomes distant. Contrastingly, the goal errors increases as soon the robot has a wider angle of perception.
These graphs also illustrates errors for observations with distance and angle variations. In both graphs, error measurements are presented in constant light conditions and without occlusion or any external noise that can affect the landmark perception.

2.2.2 Undefined Landmark Recognition

A landmark modelling is used for detecting undefined environment and frequently appearing features. The procedure is accomplished by characterising and evaluating familiarised shapes from detected objects which are characterised by sets of properties or entities. Such process is described in the following stages:

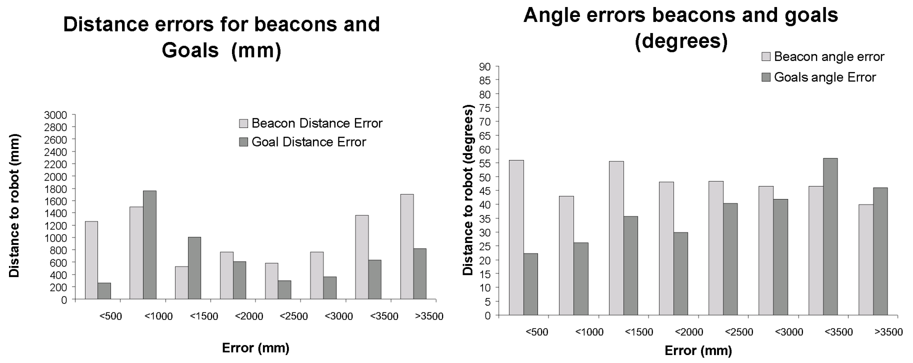


Fig. 2. Distance and angle errors in landmarks observations for beacons and goals of proposed landmark model.

- **Entity Recognition** The first stage of dynamic landmark modelling relies on feature identification from constantly observed occurrences. The information is obtained from colour surface descriptors by a landmark entity structure. An entity is integrated by pairs or triplets of blobs with unique characteristics constructed from merging and comparing linear blobs operators. The procedure interprets surface characteristics for obtaining range frequency by using the following operations:

  1. Obtain and validate entity's position from the robot's perspective.
  2. Get blobs' overlapping values with respect to their size.
  3. Evaluate compactness value from blobs situated in a bounding box.
  4. Validate eccentricity for blobs assimilated in current the entity.

- **Model Evaluation**

  The model evaluation phase describes a procedure for achieving landmark entities for a real time recognition. The process makes use of previously defined models and merges them for each sensing step. The process is described in Algorithm 1:

  From the main loop algorithm is obtained a list of candidate entities $\{E\}$ to obtain a collection of landmark models $\{L\}$. This selection requires three operations for comparing an entity with a landmark model:

  – *Colour combination* is used for checking entities with same type of colours as a landmark model.

  – *Descriptive operators*, are implemented for matching features with a similar characteristics. The matching process merges entities with a ± 0.3 range ratio from defined models.

  – *Time stamp* and *Frequency* are recognised every minute for filtering long lasting models using a removing and merging process of non leading landmark models.

  The merging process is achieved using a bubble sort comparison with a swapping stage modified for evaluating similarity values and it also eliminates 10% of the landmark

Algorithm 1 Process for creating a landmark model from a list of observed features.

```
Require: Map of observed features $\{E\}$
Require: A collection of landmark models $\{L\}$
    {The following operations generate the landmark model information.}
 1: for all $\{E\}_i \subseteq \{E\}$ do
 2:     Evaluate ColourCombination($\{E\}_i$) $\{C\}_i$
 3:     Evaluate BlobDistances($\{E\}_i$) $d_i$
 4:     Obtain TimeStamp($\{E\}_i i$) $t_i$
 5:     Create Entity($\{C\}_i$, $d_i$, $t_i$) $j$
 6:     for $\{L\}_k MATCHON \{L\}$ do {If information is similar to an achieved model }
 7:         if $j \in \{L\}_k$ then
 8:             Update $\{L\}_k(j)$ {Update modelled values and}
 9:             Increase $\{L\}_k$ frequency {Increase modelled frequency}
10:         else {If modelled information does not exist }
11:             Create $\{L\}_{k+1}(j)$ {Create model and}
12:             Increase $\{L\}_{k+1}$ frequency {Increase modelled frequency}
13:         end if
14:         if time $> 1$ min then {After one minute }
15:             MergeList($\{L\}$) {Select best models}
16:         end if
17:     end for
18: end for
```

candidates. The similarity values are evaluated using Equation 3.4 and the probability of perception using Equation 3.5:

$$p(i,j) = \frac{M(i,j)}{\sum_{k=1}^{N} M(k,j)} \tag{3.4}$$

$$M(i,j) = \sum_{l=1}^{P} E(i,j,l) \tag{3.5}$$

where N indicates the achieved candidate models, i is the sampled entity, j is the compared landmark model, $M(i,j)$ is the landmark similarity measure obtained from matching an entity's descriptors and assigning a probability of perception as described in Equation 3.6, P is the total descriptors, l is a landmark descriptor and $E(i,j,l)$ is the Euclidian distance of each landmark model compared, estimated using Equation 3.7:

$$\sum_{k=1}^{N} M(k,j) = 1 \tag{3.6}$$

$$E(i,j,l) = \sqrt{\sum_{m=1}^{L_l} \frac{(i_m - l_m)^2}{\sigma_m^2}} \tag{3.7}$$

where L_l refers to all possible operators from the current landmark model, σ_m is the standard deviation for each sampled entity i_m in a sample set and l is a landmark descriptor value.

3. Localisation Methods

Robot localisation is an environment analysis task determined by an internal state obtained from robot-environment interaction combined with any sensed observations. The traditional state assumption relies on the robot's influence over its world and on the robot's perception of its environment.
Both steps are logically divided into independent processes which use a state transition for integrating data into a predictive and updating state. Therefore, the implemented localisation methods contain measurement and control phases as part of state integration and a robot pose conformed through a Bayesian approach. On the one hand, the control phase is assigned to robot odometry which translates its motion into lateral, linear and rotational velocities. On the other hand, the measurement phase integrates robot sensed information by visual features. The following sections describe particular phase characteristics for each localisation approach.

3.1 Fuzzy Markov Method

As it is shown in the FM grid-based method of (Buschka et al., 2000) and (Herrero-Pérez et al., 2004), a grid G_t contains a number of cells for each grid element $G_t(x, y)$ for holding a probability value for a possible robot position in a range of $[0, 1]$. The fuzzy cell (*fcell*) is represented as a fuzzy trapezoid (Figure 3) defined by a tuple $< \theta, \Delta, \alpha, h, b >$, where θ is robot orientation at the trapezoid centre with values in a range of $[0, 2\pi]$; Δ is uncertainty in a robot orientation θ; h corresponds to fuzzy cell (*fcell*) with a range of $[0, 1]$; α is a slope in the trapezoid, and b is a correcting bias.

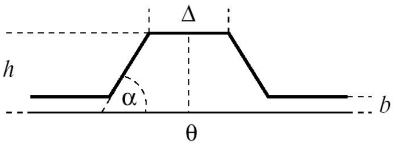


Fig. 3. Graphic representation of robot pose in an *fuzzy cell*

Since a Bayesian filtering technique is implemented, localisation process works in predict-observe-update phases for estimating robot state. In particular, the *Predict* step adjusts to motion information from robot movements. Then, the *Observe* step gathers sensed information. Finally, the *Update* step incorporates results from the *Predict* and *Observe* steps for obtaining a new estimation of a fuzzy grid-map. The process sequence is described as follows:

1. **Predict step**. During this step, robot movements along grid-cells are represented by a distribution which is continuously blurred. As described in previous work in (Herrero-Pérez et al., 2004), the blurring is based on odometry information reducing grid occupancy for robot movements as shown in Figure 4(c)). Thus, the grid state G'_t is obtained by performing a translation and rotation of G_{t-1} state distribution according to motion $\vec{u}$. Subsequently, this odometry-based blurring, B_t, is uniformly calculated for including uncertainty in a motion state.

Thus, state transition probability includes as part of robot control, the blurring from odometry values as it is described in the following equation:

$$G'_t = f(G_t \mid G_{t-1}, \vec{u}) \otimes B_t \tag{4.30}$$

2. **Observe step**. In this step, each observed landmark i is represented as a vector $\vec{z}_i$, which includes both range and bearing information obtained from visual perception. For each observed landmark $\vec{z}_i$, a grid-map $S_{i,t}$ is built such that $S_{i,t}(x, y, \theta | \vec{z}_i)$ is matched to a robot position at (x, y, θ) given an observation $\vec{r}$ at time t.
3. **Update step**. At this step, grid state G'_t obtained from the prediction step is merged with each observation step $S_{t,i}$. Afterwards, a fuzzy intersection is obtained using a product operator as follows:

$$G_t = f(z_t \mid G_t) \tag{4.31}$$

$$G_t = G'_t \times S_{t,1} \times S_{t,2} \times \cdots \times S_{t,n} \tag{4.32}$$

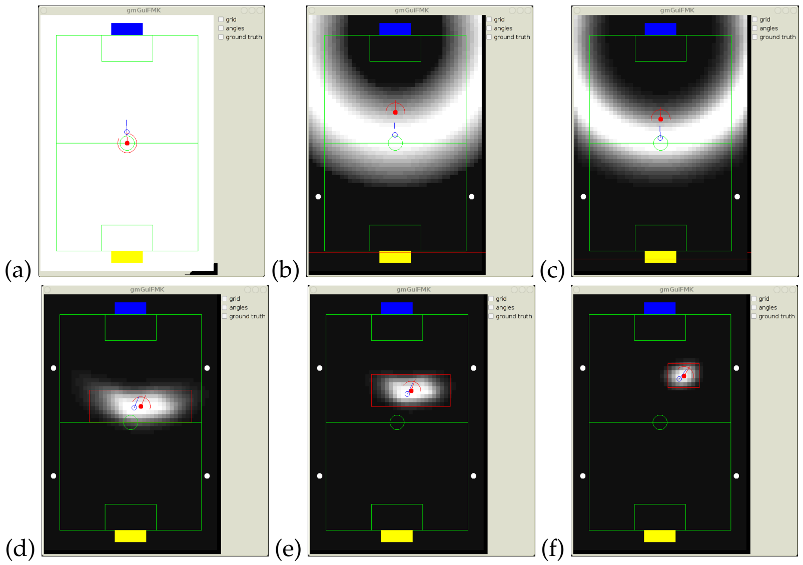


Fig. 4. In this figure is shown a simulated localisation process of FM grid starting from absolute uncertainty of robot pose (a) and some initial uncertainty (b) and (c). Through to an approximated (d) and finally to an acceptable robot pose estimation obtained from simulated environment explained in (Samperio & Hu, 2008).

A simulated example of this process is shown in Figure 4. In this set of Figures, Figure 4(a) illustrates how the system is initialised with absolute uncertainty of robot pose as the white areas. Thereafter, the robot incorporates landmark and goal information where each grid state G_t is updated whenever an observation can be successfully matched to a robot position, as

illustrated in Figure 4(b). Subsequently, movements and observations of various landmarks enable the robot to localise itself, as shown from Figure 4(c) to Figure 4(f).
This method's performance is evaluated in terms of accuracy and computational cost during a real time robot execution. Thus, a reasonable *fcell* size of 20 cm^2 is addressed for less accuracy and computing cost in a pitch space of 500cmx400cm.
This localisation method offers the following advantages, according to (Herrero-Pérez et al., 2004):

- Fast recovery from previous errors in the robot pose estimation and kidnappings.
- Multi-hypotheses for robot pose (x, y) .
- It is much faster than classical Markovian approaches.

However, its disadvantages are:

- Mono-hypothesis for orientation estimation.
- It is very sensitive to sensor errors.
- The presence of false positives makes the method unstable in noisy conditions.
- Computational time can increase dramatically.

3.2 Extended Kalman Filter method

Techniques related to EKF have become one of the most popular tools for state estimation in robotics. This approach makes use of a state vector for robot positioning which is related to environment perception and robot odometry. For instance, robot position is adapted using a vector s_t which contains (x, y) as robot position and θ as orientation.

$$\mathbf{s} = \begin{pmatrix} x_{robot} \\ y_{robot} \\ \theta_{robot} \end{pmatrix} \tag{4.17}$$

As a Bayesian filtering method, EKF is implemented *Predict* and *Update* steps, described in detail below:
1. Prediction step. This phase requires of an initial state or previous states and robot odometry information as control data for predicting a state vector. Therefore, the current robot state s_t^- is affected by odometry measures, including a noise approximation for error and control estimations P_t^-. Initially, robot control probability is represented by using:

$$s_t^- = f(s_{t-1}, u_{t-1}, w_t) \tag{4.18}$$

where the nonlinear function f relates the previous state s_{t-1}, control input u_{t-1} and the process noise w_t.
Afterwards, a covariance matrix P_t^- is used for representing errors in state estimation obtained from the previous step's covariance matrix P_{t-1} and defined process noise. For that reason, the covariance matrix is related to the robot's previous state and the transformed control data, as described in the next equation:

$$P_t^- = A_t P_{t-1} A_t^T + W_t Q_{t-1} W_t^T \tag{4.19}$$

where $A_t P_{t-1} A_t^T$ is a progression of P_{t-1} along a new movement and A_t is defined as follows:

$$A_t = fs = \begin{pmatrix} 1 & 0 & -u_t^{lat} cos\theta_t - u_t^{lin} sen\theta_{t-1} \\ 0 & 1 & u_t^{lin} cos\theta_t - u_t^{lat} sen\theta_{t-1} \\ 0 & 0 & 1 \end{pmatrix} \tag{4.19}$$

and $W_t Q_{t-1} W_t^T$ represents odometry noise, W_t is Jacobian motion state approximation and Q_t is a covariance matrix as follows:

$$Q_t = E[w_t w_t^T] \tag{4.20}$$

The Sony AIBO robot may not be able to obtain a landmark observation at each localisation step but it is constantly executing a motion movement. Therefore, it is assumed that frequency of odometry calculation is higher than visually sensed measurements. For this reason, control steps are executed independently from measurement states (Kiriy & Buehler, 2002) and covariance matrix actualisation is presented as follows:

$$s_t = s_t^- \tag{4.21}$$

$$P_t = P_t^- \tag{4.22}$$

2. Updating step. During this phase, sensed data and noise covariance P_t are used for obtaining a new state vector. The sensor model is also updated using measured landmarks $m_{1\cdots 6,(x,y)}$ as environmental descriptive data. Thus, each z_t^i of the i landmarks is measured as distance and angle with a vector (r_t^i, ϕ_t^i). In order to obtain an updated state, the next equation is used:

$$s_t = s_{t-1} + K_t^i (z_t^i - \hat{z}_t^i) = s_{t-1} + K_t^i (z_t^i - h^i(s_{t-1})) \tag{4.23}$$

where $h^i(s_{t-1})$ is a predicted measurement calculated from the following non-linear functions:

$$\hat{z}_t^i = h^i(s_{t-1}) = \begin{pmatrix} \sqrt{(m_{t,x}^i - s_{t-1,x})^2 + (m_{t,y}^i - s_{t-1,y})} \\ atan^2(m_{t,x}^i - s_{t-1,x}, m_{t,y}^i - s_{t-1,y}) - s_{t-1,\theta} \end{pmatrix} \tag{4.24}$$

Then, the *Kalman gain*, K_t^i, is obtained from the next equation:

$$K_t^i = P_{t-1} (H_t^i)^T (S_t^i)^{-1} \tag{4.25}$$

where S_t^i is the uncertainty for each predicted measurement $\hat{z}_t^i$ and is calculated as follows:

$$S_t^i = H_t^i P_{t-1} (H_t^i)^T + R_t^i \tag{4.26}$$

Then H_t^i describes changes in the robot position as follows:

$$\begin{aligned} H_t^i &= h^i(s_{t-1}) s_t \\ &= \begin{pmatrix} -\frac{m_{t,x}^i - s_{t-1,x}}{\sqrt{(m_{t,x}^i - s_{t-1,x})^2 + (m_{t,y}^i - s_{t-1,y})^2}} & -\frac{m_{t,y}^i - s_{t-1,y}}{\sqrt{(m_{t,x}^i - s_{t-1,x})^2 + (m_{t,y}^i - s_{t-1,y})^2}} & 0 \\ \frac{m_{t,y}^i - s_{t-1,y}}{(m_{t,x}^i - s_{t-1,x})^2 + (m_{t,y}^i - s_{t-1,y})^2} & -\frac{m_{t,x}^i - s_{t-1,x}}{(m_{t,x}^i - s_{t-1,x})^2 + (m_{t,y}^i - s_{t-1,y})^2} & -1 \\ 0 & 0 & 0 \end{pmatrix} \end{aligned} \tag{4.27}$$

where R_t^i represents the measurement noise which was empirically obtained and P_t is calculated using the following equation:

$$P_t = (I - K_t^i H_t^i) P_{t-1} \tag{4.28}$$

Finally, as not all $\hat{z}_t^i$ values are obtained at every observation, z_t^i values are evaluated for each observation and δ_t^i is a confidence measurement obtained from Equation (4.29). The confidence observation measurement has a threshold value between 5 and 100, which varies according to localisation quality.

$$\delta_t^i = (z_t^i - \hat{z}_t^i)^T (S_t^i)^{-1} (z_t^i - \hat{z}_t^i) \tag{4.29}$$

3.3 FM-EKF method

Merging the FM and EKF algorithms is proposed in order to achieve computational efficiency, robustness and reliability for a novel robot localisation method. In particular, the FM-EKF method deals with inaccurate perception and odometry data for combining method hypotheses in order to obtain the most reliable position from both approaches.
The hybrid procedure is fully described in Algorithm 2, in which the *fcell* grid size is (50-100 cm) which is considerably wider than FM's. Also the *fcell* is initialised in the space map centre. Subsequently, a variable is iterated for controlling FM results and it is used for comparing robot EKF positioning quality. The localisation quality indicates if EKF needs to be reset in the case where the robot is lost or the EKF position is out of FM range.

Algorithm 2 Description of the FM-EKF algorithm.

```
Require: $position_{FM}$ over all pitch
Require: $position_{EKF}$ over all pitch
 1: while $robotLocalise$ do
 2:     $\{Execute''Predict''phasesforFMandEKF\}$
 3:     Predict $position_{FM}$ using motion model
 4:     Predict $position_{EKF}$ using motion model
 5:     $\{Execute''Correct''phasesforFMandEKF\}$
 6:     Correct $position_{FM}$ using perception model
 7:     Correct $position_{EKF}$ using perception model
 8:     $\{Checkqualityoflocalisationfor EKFusingFM\}$
 9:     if $(quality(position_{FM}) \gg quality(position_{EKF})$ then
10:         Initialise $position_{EKF}$ to $position_{FM}$
11:     else
12:         robot position $\leftarrow position_{EKF}$
13:     end if
14: end while
```

The FM-EKF algorithm follows the predict-observe-update scheme as part of a Bayesian approach. The input data for the algorithm requires similar motion and perception data. Thus, the hybrid characteristics maintain a combined hypothesis of robot pose estimation using data that is independently obtained. Conversely, this can adapt identical measurement and control information for generating two different pose estimations where, under controlled circumstances one depends on the other.
From one viewpoint, FM localisation is a robust solution for noisy conditions. However, it is also computationally expensive and cannot operate efficiently in real-time environments

with a high resolution map. Therefore, its computational accuracy is inversely proportional to the *fcell* size. From a different perspective, EKF is an efficient and accurate positioning system which can converge computationally faster than FM. The main drawback of EKF is a misrepresentation in the multimodal positioning information and method initialisation.

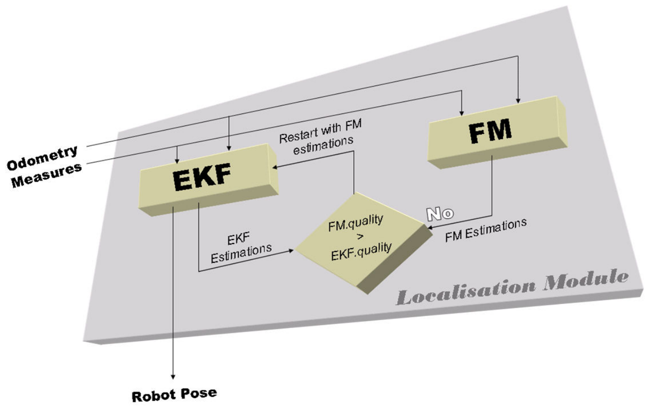


Fig. 5. Flux diagram of hybrid localisation process.

The hybrid method combines FM grid accuracy with EKF tracking efficiency. As it is shown in Figure 5, both methods use the same control and measurement information for obtaining a robot pose and positioning quality indicators. The EKF quality value is originated from the eigenvalues of the error covariance matrix and from noise in the grid- map.
As a result, EKF localisation is compared with FM quality value for obtaining a robot pose estimation. The EKF position is updated whenever the robot position is lost or it needs to be initialised. The FM method runs considerably faster though it is less accurate.
This method implements a Bayesian approach for robot-environment interaction in a localisation algorithm for obtaining robot position and orientation information. In this method a wider *fcell* size is used for the FM grid-map implementation and EKF tracking capabilities are developed to reduce computational time.

4. System Overview

The configuration of the proposed HRI is presented in Figure 6. The user-robot interface manages robot localisation information, user commands from a GUI and the overhead tracking, known as the VICON tracking system for tracking robot pose and position. This overhead tracking system transmits robot heading and position data in real time to a GUI where the information is formatted and presented to the user.
The system also includes a robot localisation as a subsystem composed of visual perception, motion and behaviour planning modules which continuously emits robot positioning information. In this particular case, localisation output is obtained independently of robot behaviour moreover they share same processing resources. Additionally, robot-visual information can be generated online from GUI from characterising environmental landmarks into robot configuration.
Thus, the user can manage and control the experimental execution using online GUI tasks. The GUI tasks are for designing and controlling robot behaviour and localisation methods,

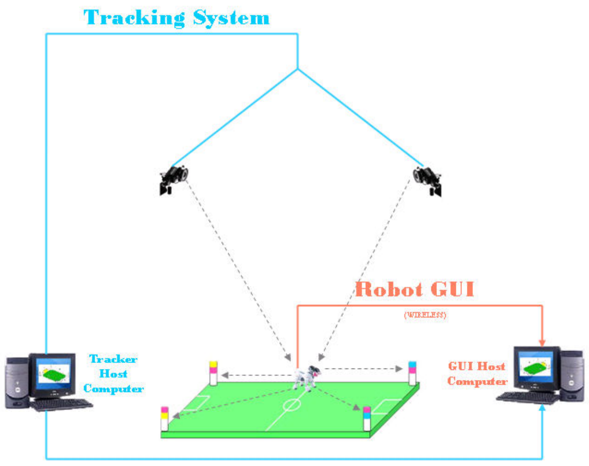


Fig. 6. Complete configuration of used human-robot interface.

and for managing simulated and experimental results. Moreover, tracking results are the experiments' input and output of a grand truth that is evaluating robot self-localisation results.

5. Experimental Results

The presented experimental results contain the analysis of the undefined landmark models and a comparison of implemented localisation methods. The undefined landmark modelling is designed for detecting environment features that could support the quality of the localisation methods. All localisation methods make use of defined landmarks as main source of information.
The first set of experiments describe the feasibility for employing a not defined landmark as a source for localisation. These experiments measure the robot ability to define a new landmark in an indoor but dynamic environment. The second set of experiments compare the quality of localisation for the FM, EKF and FM-EKF independently from a random robot behaviour and environment interaction. Such experiments characterise particular situations when each of the methods exhibits an acceptable performance in the proposed system.

5.1 Dynamic landmark acquisition

The performance for angle and distance is evaluated in three experiments. For the first and second experiments, the robot is placed in a fixed position on the football pitch where it continuously pans its head. Thus, the robot maintains simultaneously a perception process and a dynamic landmark creation. Figure 7 show the positions of 1683 and 1173 dynamic models created for the first and second experiments over a duration of five minutes.
Initially, newly acquired landmarks are located at 500 mm and with an angle of $3\Pi/4rad$ from the robot's centre. Results are presented in Table **??**. The tables for Experiments 1 and 2, illustrate the mean ($\tilde{x}$) and standard deviation (σ) of each entity's distance, angle and errors from the robot's perspective.
In the third experiment, landmark models are tested during a continuous robot movement. This experiment consists of obtaining results at the time the robot is moving along a circular

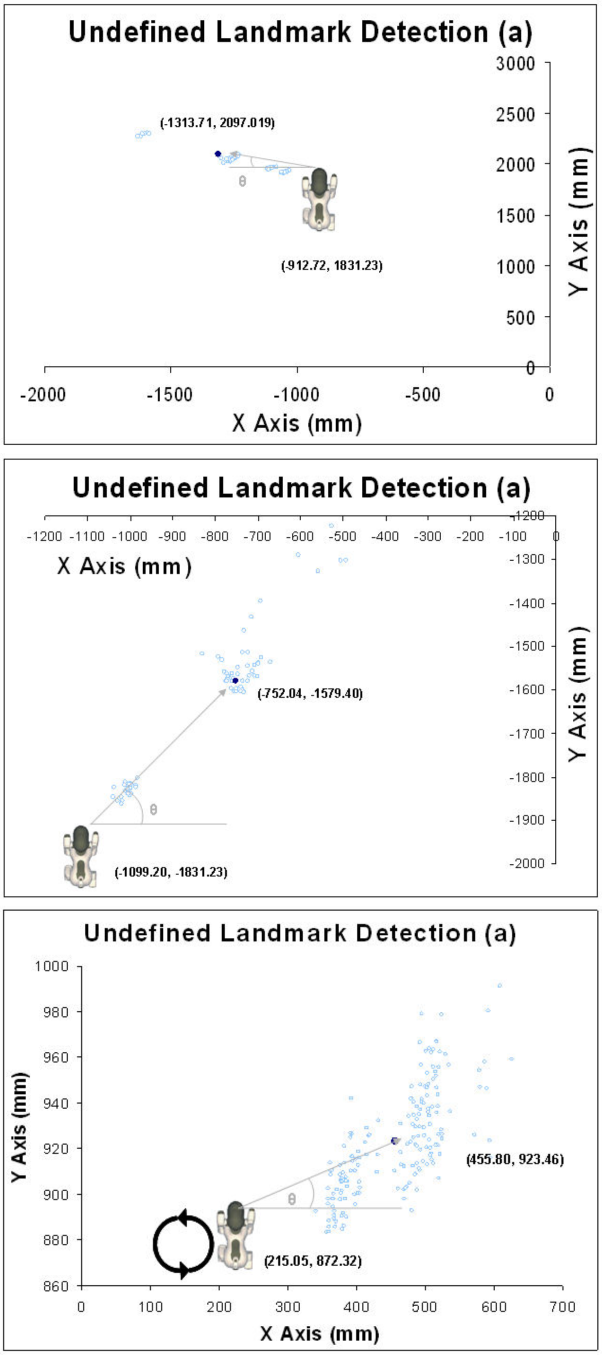


Fig. 7. Landmark model recognition for Experiments 1, 2 and 3

Experpiment 1	**Distance**	**Angle**	**Error in Distance**	**Error in Angle**
Mean	489.02	146.89	256.46	2.37
StdDev	293.14	9.33	133.44	8.91

Experpiment 2	**Distance**	**Angle**	**Error in Distance**	**Error in Angle**
Mean	394.02	48.63	86.91	2.35
StdDev	117.32	2.91	73.58	1.71

Experpiment 3	**Distance**	**Angle**	**Error in Distance**	**Error in Angle**
Mean	305.67	12.67	90.30	3.61
StdDev	105.79	4.53	54.37	2.73

Table 2. Mean and standard deviation for experiment 1, 2 and 3.

trajectory with 20 cm of bandwidth radio, and whilst the robot's head is continuously panning. The robot is initially positioned 500 mm away from a coloured beacon situated at 0 *degrees* from the robot's mass centre. The robot is also located in between three defined and one undefined landmarks. Results obtained from dynamic landmark modelling are illustrated in Figure 7. All images illustrate the generated landmark models during experimental execution. Also it is shown darker marks on all graphs represent an accumulated average of an observed landmark model.
This experiment required 903 successful landmark models detected over five minute duration of continuous robot movement and the results are presented in the last part of the table for Experiment 3. The results show magnitudes for mean ($\tilde{x}$) and standard deviation (σ), distance, angle and errors from the robot perspective.
Each of the images illustrates landmark models generated during experimental execution, represented as the accumulated average of all observed models. In particular for the first two experiments, the robot is able to offer an acceptable angular error estimation in spite of a variable proximity range. The results for angular and distance errors are similar for each experiment. However, landmark modelling performance is susceptible to perception errors and obvious proximity difference from the perceived to the sensed object.
The average entity of all models presents a minimal angular error in a real-time visual process. An evaluation of the experiments is presented in Box and Whisker graphs for error on position, distance and angle in Figure 8.
Therefore, the angle error is the only acceptable value in comparison with distance or positioning performance. Also, the third experiment shows a more comprehensive real-time measuring with a lower amount of defined landmark models and a more controllable error performance.

5.2 Comparison of localisation methods

The experiments were carried out in three stages of work: (i) simple movements; (ii) combined behaviours; and (iii) kidnapped robot. Each experiment set is to show robot positioning abilities in a RoboCup environment. The total set of experiment updates are of 15, with 14123 updates in total. In each experimental set, the robot poses estimated by EKF, FM and FM-EKF localisation methods are compared with the ground truth generated by the overhead vision system. In addition, each experiment set is compared respectively within its processing time. Experimental sets are described below:

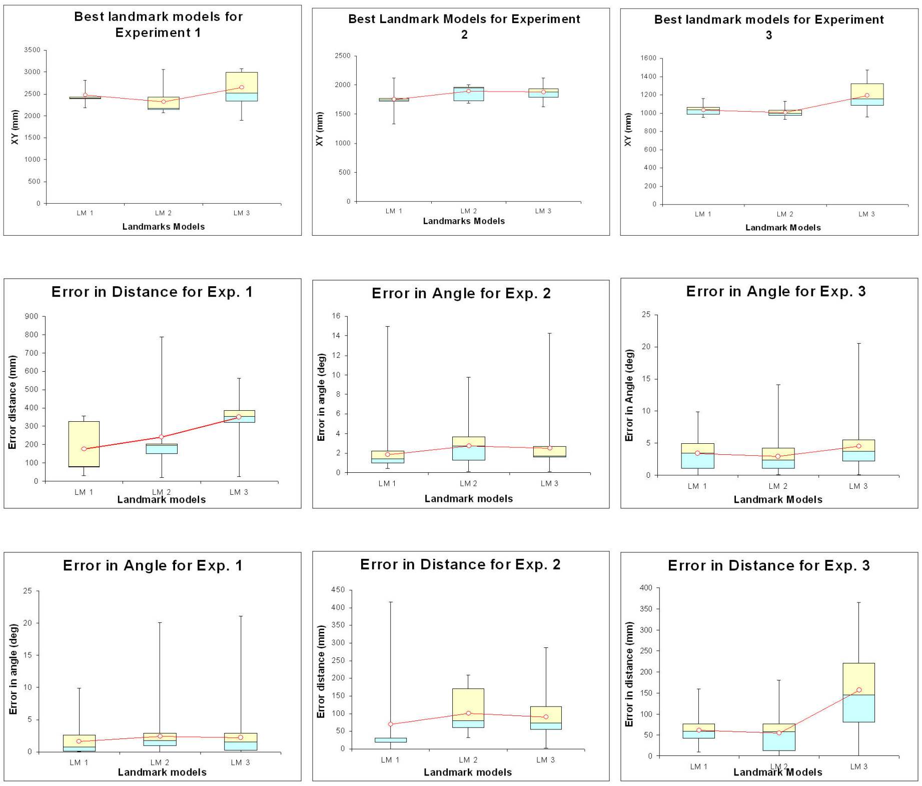


Fig. 8. Error in angle for Experiments 1, 2 and 3.

1. **Simple Movements**. This stage includes straight and circular robot trajectories in forward and backward directions within the pitch.
2. **Combined Behaviour**. This stage is composed by a pair of high level behaviours. Our first experiment consists of reaching a predefined group of coordinated points along the pitch. Then, the second experiment is about playing alone and with another dog to obtain localisation results during a long period.
3. **Kidnapped Robot**. This stage is realised randomly in sequences of kidnap time and pose. For each kidnap session the objective is to obtain information about where the robot is and how fast it can localise again.

All experiments in a playing session with an active localisation are measured by showing the type of environment in which each experiment is conducted and how they directly affect robot behaviour and localisation results. In particular, the robot is affected by robot displacement, experimental time of execution and quantity of localisation cycles. These characteristics are described as follows and it is shown in Table 3:

1. **Robot Displacement** is the accumulated distance measured from each simulated method step from the perspective of the grand truth mobility.

2. **Localisation Cycles** include any completed iteration from update-observe-predict stages for any localisation method.
3. **Time of execution** refers to total amount of time taken for each experiment with a time of 1341.38 s for all the experiments.

	Exp. 1	Exp. 2	Exp. 3
Displacement (mm)	15142.26	5655.82	11228.42
Time of Execution (s)	210.90	29.14	85.01
Localisation Cycles (iterations)	248	67	103

Table 3. Experimental conditions for a simulated environment.

The experimental output depends on robot behaviour and environment conditions for obtaining parameters of performance. On the one side, robot behaviour is embodied by the specific robot tasks executed such as *localise*, *kick the ball*, *search for the ball*, *search for landmarks*, *search for players*, *move to a point in the pitch*, *start*, *stop*, *finish*, and so on. On the other side, robot control characteristics describe robot performance on the basis of values such as: *robot displacement*, *time of execution*, *localisation cycles* and *landmark visibility*. Specifically, robot performance criteria are described for the following environment conditions:

1. **Robot Displacement** is the distance covered by the robot for a complete experiment, obtained from grand truth movement tracking. The total displacement from all experiments is 146647.75 mm.
2. **Landmark Visibility** is the frequency of the detected true positives for each landmark model among all perceived models. Moreover, the visibility ranges are related per each localisation cycle for all natural and artificial landmarks models. The average landmark visibility obtained from all the experiments is in the order of 26.10 % landmarks per total of localisation cycles.
3. **Time of Execution** is the time required to perform each experiment. The total time of execution for all the experiments is 902.70 s.
4. **Localisation Cycles** is a complete execution of a correct and update steps through the localisation module. The amount of tries for these experiments are 7813 cycles.

The internal robot conditions is shown in Table **??**:

	Exp 1	Exp 2	Exp 3
Displacement (mm)	5770.72	62055.79	78821.23
Landmark Visibility (true positives/total obs)	0.2265	0.3628	0.2937
Time of Execution (s)	38.67	270.36	593.66
Localisation Cycles (iterations)	371	2565	4877

Table 4. Experimental conditions for a real-time environment.

In Experiment 1, the robot follows a trajectory in order to localise and generate a set of visible ground truth points along the pitch. In Figures 9 and 10 are presented the error in X and Y axis by comparing the EKF, FM, FM-EKF methods with a grand truth. In this graphs it is shown a similar performance between the methods EKF and FM-EKF for the error in X and Y

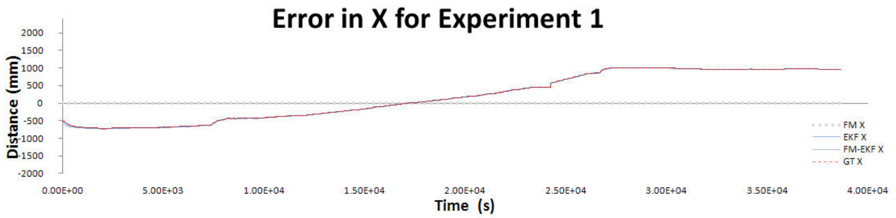


Fig. 9. Error in X axis during a simple walk along the pitch in Experiment 1.

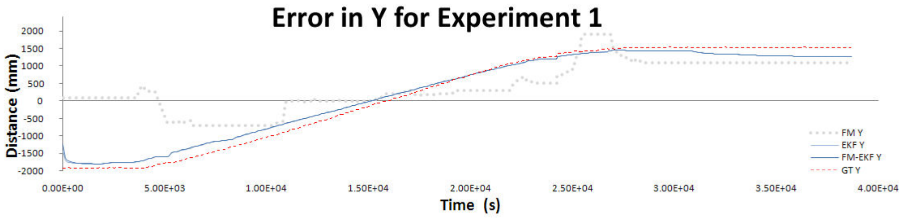


Fig. 10. Error in Y axis during a simple walk along the pitch in Experiment 1.

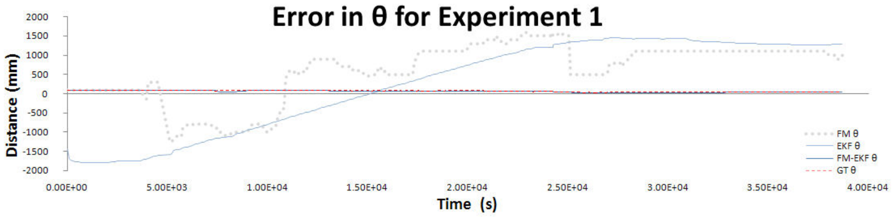


Fig. 11. Error in θ axis during a simple walk along the pitch in Experiment 1.

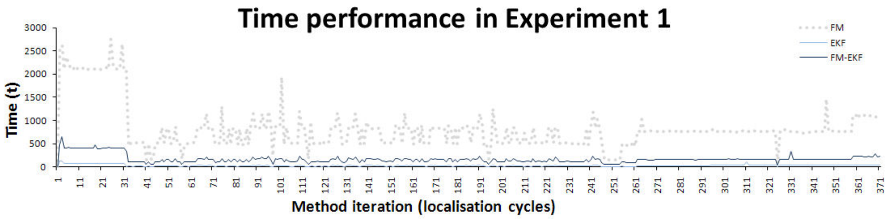


Fig. 12. Time performance for localisation methods during a walk along the pitch in Exp. 1.

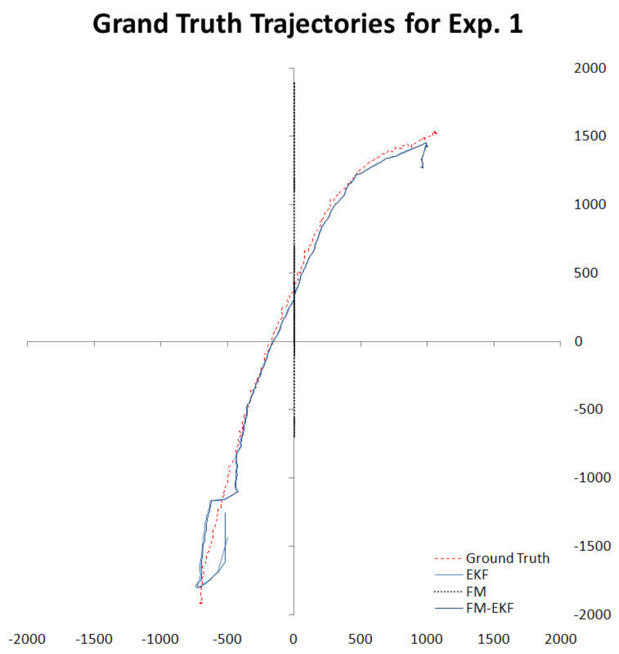


Fig. 13. Robot trajectories for EKF, FM, FM-EKF and the overhead camera in Exp. 1.

axis but a poor performance of the FM. However for the orientation error displayed in Figure 11 is shown that whenever the robot walks along the pitch without any lack of information, FM-EKF improves comparatively from the others. Figure 12 shows the processing time for all methods, in which the proposed FM-EKF method is faster than the FM method, but slower than the EKF method. Finally, in Figure 13 is presented the estimated trajectories and the overhead trajectory. As can be seen, during this experiment is not possible to converge accurately for FM but it is for EKF and FM-EKF methods where the last one presents a more accurate robot heading.
For Experiment 2, is tested a combined behaviour performance by evaluating a playing session for a single and multiple robots. Figures 14 and 15 present as the best methods the EKF and FM-EKF with a slightly improvement of errors in the FM-EKF calculations. In Figure 16 is shown the heading error during a playing session where the robot visibility is affected by a constantly use of the head but still FM-EKF, maintains an more likely performance compared to the grand truth. Figure 17 shows the processing time per algorithm iteration during the robot performance with a faster EKF method. Finally, Figure 18 shows the difference of robot trajectories estimated by FM-EKF and overhead tracking system.
In the last experiment, the robot was randomly kidnapped in terms of time, position and orientation. After the robot is manually deposited in a different pitch zone, its localisation performance is evaluated and shown in the figures for Experiment 3. Figures 19 and 20 show positioning errors for X and Y axis during a kidnapping sequence. Also, FM-EKF has a similar development for orientation error as it is shown in Figure 21. Figure 22 shows the processing

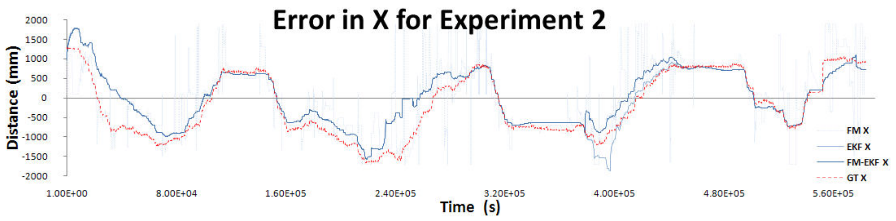


Fig. 14. Error in X axis during a simple walk along the pitch in Experiment 2.

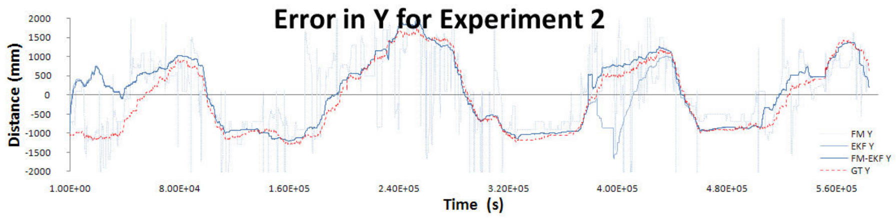


Fig. 15. Error in Y axis during a simple walk along the pitch in Experiment 2.

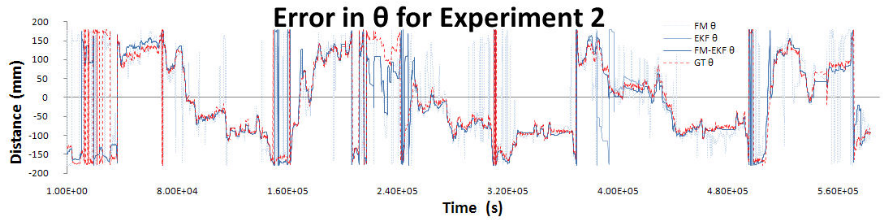


Fig. 16. Error in θ axis during a simple walk along the pitch in Experiment 2.

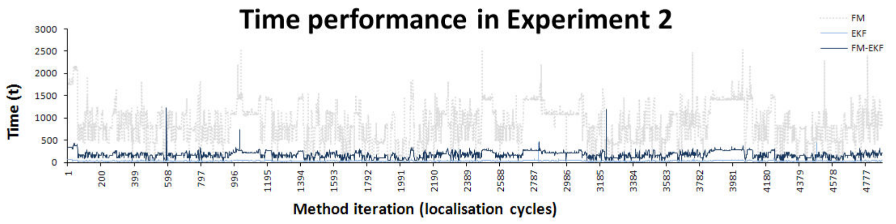


Fig. 17. Time performance for localisation methods during a walk along the pitch in Exp. 2.

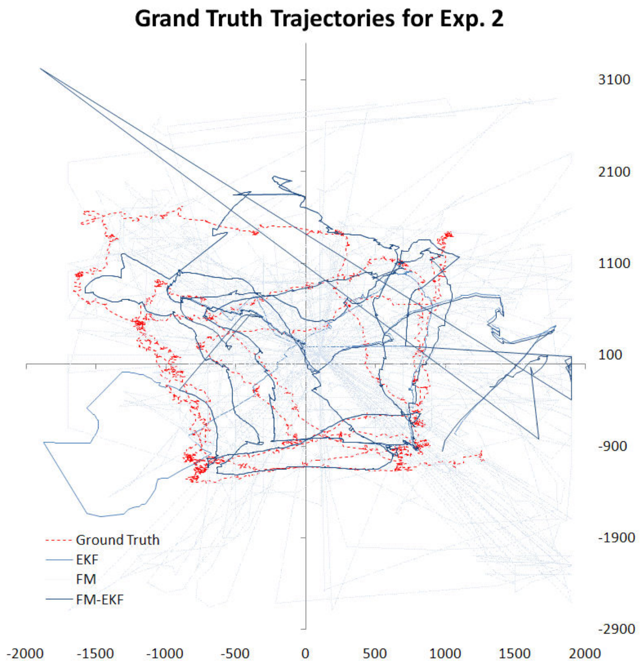


Fig. 18. Robot trajectories for EKF, FM, FM-EKF and the overhead camera in Exp. 2.

time per iteration for all algorithms a kidnap session. Finally, in Figure 23 and for clarity reasons is presented the trajectories estimated only by FM-EKF, EKF and overhead vision system. Results from kidnapped experiments show the resetting transition from a local minimum to fast convergence in 3.23 seconds. Even EKF has the most efficient computation time, FM-EKF offers the most stable performance and is a most suitable method for robot localisation in a dynamic indoor environment.

6. Conclusions

This chapter presents an implementation of real-time visual landmark perception for a quadruped walking robot in the RoboCup domain. The proposed approach interprets an object by using symbolic representation of environmental features such as natural, artificial or undefined. Also, a novel hybrid localisation approach is proposed for fast and accurate robot localisation of an active vision platform. The proposed FM-EKF method integrates FM and EKF algorithms using both visual and odometry information.

The experimental results show that undefined landmarks can be recognised accurately during static and moving robot recognition sessions. On the other hand, it is clear that the hybrid method offers a more stable performance and better localisation accuracy for a legged robot which has noisy odometry information. The distance error is reduced to ±20 mm and the orientation error is 0.2 degrees.

Further work will focus on adapting for invariant scale description during real time image processing and of optimising the filtering of recognized models. Also, research will focus on

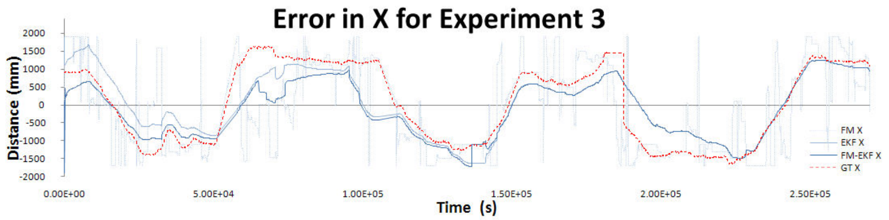


Fig. 19. Error in X axis during a simple walk along the pitch in Experiment 3.

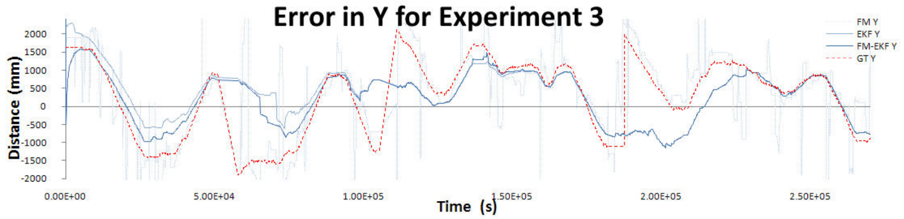


Fig. 20. Error in Y axis during a simple walk along the pitch in Experiment 3.

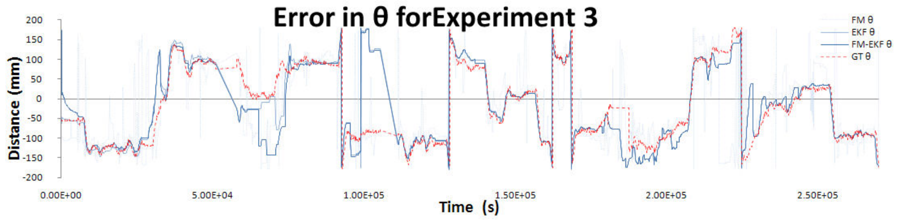


Fig. 21. Error in θ axis during a simple walk along the pitch in Experiment 3.

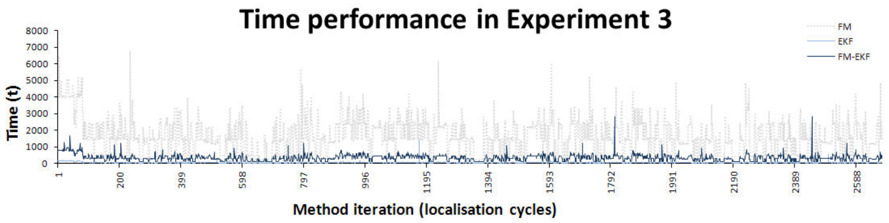


Fig. 22. Time performance for localisation methods during a walk along the pitch in Exp. 3.

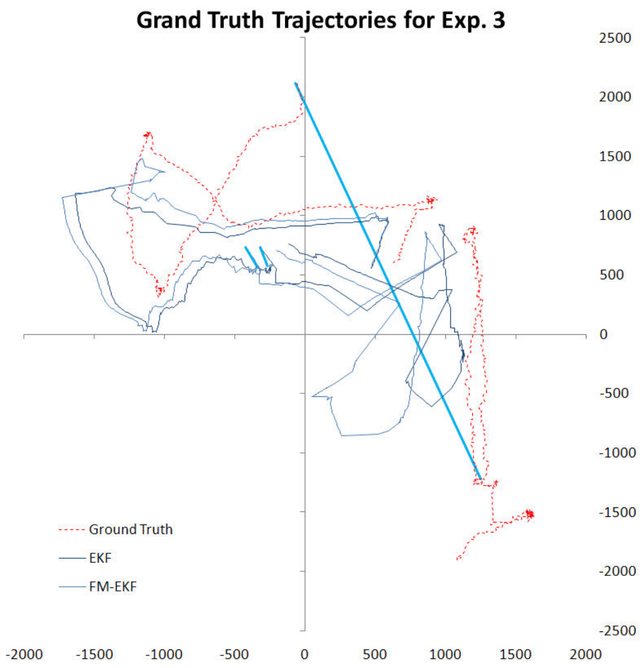


Fig. 23. Robot trajectories for EKF, FM, FM-EKF and the overhead camera in Exp. 3., where the thick line indicates kidnapped period.

the reinforcement of the quality in observer mechanisms for odometry and visual perception, as well as the improvement of landmark recognition performance.

7. Acknowledgements

We would like to thank TeamChaos (http://www.teamchaos.es) for their facilities and programming work and Essex technical team (http://essexrobotics.essex.ac.uk) for their support in this research. Part of this research was supported by a CONACyT (Mexican government) scholarship with reference number 178622.

8. References

Baltzakis, H. & Trahanias, P. (2002). Hybrid mobile robot localization using switching state-space models., *Proc. of IEEE International Conference on Robotics and Automation*, Washington, DC, USA, pp. 366–373.

Buschka, P., Saffiotti, A. & Wasik, Z. (2000). Fuzzy landmark-based localization for a legged robot, *Proc. of IEEE Intelligent Robots and Systems*, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1205 – 1210.

Duckett, T. & Nehmzow, U. (2000). Performance comparison of landmark recognition systems for navigating mobile robots, *Proc. 17th National Conf. on Artificial Intelligence (AAAI-2000)*, AAAI Press/The MIT Press, pp. 826 – 831.

Fox, D., Burgard, W., Dellaert, F. & Thrun, S. (1999). Monte carlo localization: Efficient position estimation for mobile robots, *AAAI/IAAI*, pp. 343–349.
URL: *http://citeseer.ist.psu.edu/36645.html*

Fox, D., Burgard, W. & Thrun, S. (1998). Markov localization for reliable robot navigation and people detection, *Sensor Based Intelligent Robots*, pp. 1–20.

Gutmann, J.-S. (2002). Markov-kalman localization for mobile robots., *ICPR (2)*, pp. 601–604.

Gutmann, J.-S., Burgard, W., Fox, D. & Konolige, K. (1998). An experimental comparison of localization methods, *Proc. of IEEE/RSJ International Conference on Intelligen Robots and Systems*, Vol. 2, pp. 736 – 743.

Hatice, K., C., B. & A., L. (2006). Comparison of localization methodsfor a robot soccer team, *International Journal of Advanced Robotic Systems* **3**(4): 295–302.

Hayet, J., Lerasle, F. & Devy, M. (2002). A visual landmark framework for indoor mobile robot navigation, *IEEE International Conference on Robotics and Automation*, Vol. 4, pp. 3942 – 3947.

Herrero-Pérez, D., Martínez-Barberá, H. M. & Saffiotti, A. (2004). Fuzzy self-localization using natural features in the four-legged league, *in* D. Nardi, M. Riedmiller & C. Sammut (eds), *RoboCup 2004: Robot Soccer World Cup VIII*, LNAI, Springer-Verlag, Berlin, DE. Online at http://www.aass.oru.se/~asaffio/.

Jensfelt, P., Austin, D., Wijk, O. & Andersson, M. (2000). Feature based condensation for mobile robot localization, *Proc. of IEEE International Conference on Robotics and Automation*, pp. 2531 – 2537.

Kiriy, E. & Buehler, M. (2002). Three-state extended kalman filter for mobile robot localization, *Technical Report TR-CIM 05.07*, McGill University, Montreal, Canada.

Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I. & Osawa, E. (1997). Robocup: The robot world cup initiative, *International Conference on Autonomous Agents archive*, Marina del Rey, California, United States, pp. 340 – 347.

Kristensen, S. & Jensfelt, P. (2003). An experimental comparison of localisation methods, *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 992 – 997.

Luke, R., Keller, J., Skubic, M. & Senger, S. (2005). Acquiring and maintaining abstract landmark chunks for cognitive robot navigation, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2566– 2571.

Maosen, W., Hashem, T. & Zell, A. (2005). Robot navigation using biosonar for natural landmark tracking, *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp. 3 – 7.

Quoc, V. D., Lozo, P., Jain, L., Webb, G. I. & Yu, X. (2004). A fast visual search and recognition mechanism for real-time robotics applications, *Lecture notes in computer science* **3339**(17): 937–942. XXII, 1272 p.

Samperio, R. & Hu, H. (2008). An interactive HRI for walking robots in robocup, *In Proc. of the International Symposium on Robotics and Automation IEEE*, ZhangJiaJie, Hunan, China.

Samperio, R. & Hu, H. (2010). Implementation of a localisation-oriented HRI for walking robots in the robocup environment, *International Journal of Modelling, Identification and Control (IJMIC)* **12**(2).

Samperio, R., Hu, H., Martin, F. & Mantellan, V. (2007). A hybrid approach to fast and accurate localisation for legged robots, *Robotica* . Cambridge Journals (In Press).

Sung, J. A., Rauh, W. & Recknagel, M. (1999). Circular coded landmark for optical 3d-measurement and robot vision, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 2, pp. 1128 – 1133.

Tanaka, K., Kimuro, Y., Okada, N. & Kondo, E. (2004). Global localization with detection of changes in non-stationary environments, *Proc. of IEEE International Conference on Robotics and Automation*, pp. 1487 – 1492.

Thrun, S., Beetz, M., Bennewitz, M., Burgard, W., Cremers, A., Dellaert, F., Fox, D., Hahnel, D., Rosenberg, C., Roy, N., Schulte, J. & Schulz, D. (2000). Probabilistic algorithms and the interactive museum tour-guide robot minerva, *International Journal of Robotics Research* **19**(11): 972–999.

Thrun, S., Fox, D., Burgard, W. & Dellaert, F. (2001). Robust monte carlo localization for mobile robots, *Journal of Artificial Intelligence* **128**(1-2): 99–141.
URL: *http://citeseer.ist.psu.edu/thrun01robust.html*

Watman, C., Austin, D., Barnes, N., Overett, G. & Thompson, S. (2004). Fast sum of absolute differences visual landmark, *IEEE International Conference on Robotics and Automation*, Vol. 5, pp. 4827 – 4832.

Yoon, K. J. & Kweon, I. S. (2001). Artificial landmark tracking based on the color histogram, *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, Vol. 4, pp. 1918–1923.

Ranging fusion for accurating state of the art robot localization

Hamed Bastani[1] and Hamid Mirmohammad-Sadeghi[2]
[1]*Jacobs University Bremen, Germany*
[2]*Isfahann University of Technology, Iran*

1. Introduction

Generally speaking, positioning and localization give somehow the same comprehension in terminology. They can be defined as a mechanism for realizing the spatial relationship between desired features. Independent from the mechanisms themselves, they all have certain requirements to fulfil. Scale of measurements and granularity is one important aspect to be investigated. There are limitations, and on the other hand expectations, depending on each particular application. Accuracy gives the closeness of the estimated solution with respect to the associated real position of a feature in the work space (a.k.a ground truth position). Consistency of the realized solution and the ground truth, is represented by precision. Other parameters are still existing which leave more space for investigation depending on the technique used for localization, parameters such as refresh rate, cost (power consumption, computation, price, infrastructure installation burden, ...), mobility and adaptively to the environment (indoor, outdoor, space robotics, underwater vehicles, ...) and so on.
From the mobile robotics perspective, localization and mapping are deeply correlated. There is the whole field of Simultaneous Localization and Mapping (SLAM), which deals with the employment of local robot sensors to generate good position estimates and maps; see (Thrun, 2002) for an overview. SLAM is also intensively studied from the multi robot perspective. This is while SLAM requires high end obstacle detection sensors such as laser range finders and it is computationally quite expensive.
Aside from SLAM, there are state of the art positioning techniques which can be anyhow fused in order to provide higher accuracy, faster speed and to be capable of dealing with systems with higher degrees of complexity. Here, the aim is to first of all survey an introductory overview on the common robot localization techniques, and particularly then focus on those which employ a rather different approach, i.e ranging by means of specially radio wireless technology. It will be shortly seen that mathematical tools and geometrical representation of a graph model containing multiple robots as the network nodes, can be considered a feature providing higher positioning accuracy compared to the traditional methods. Generally speaking, such improvements are scalable and also provide robustness against variant inevitable disturbing environmental features and measurement noise.

2. State of the Art Robot Localization

Localization in the field of mobile robotics is vast enough to fall into unique judgements and indeed categorization. There are plenty of approaches tackling the problem from different perspectives. In order to provide a concrete understanding of grouping, we start facing the localization mechanisms with a form of categorization as follows.

- *Passive Localization*: where already present signals are observed and processed by the system in order to deduce location of desired features. Clearly, depending on the signal's specifications, special sensory system as well as available computation power, certain flexibility is required due to passiveness.

- *Active Localization*: in which, the localization mechanism itself generates and uses its own signals for the localization purpose.

Preference is up to the particular application where it may choose a solution which falls in either of the above classes. However, a surface comparison says the second approach would be more environmental independent and therefore, more reliable for a wider variety of applications. This again will be a tradeoff for some overhead requirements such as processing power, computation resources and possibly extra auxiliary hardware subunits. From another side of assessment, being *utilized hardware* point of view, we can introduce the techniques below, which each itself is either passive or active:

1. *Dead reckoning*: uses encoders, principally to realize translational movements from rotational measurements based on integration. Depending on the application, there are different resolutions defined. This class is the most basic but at the same time the most common one, applied in mobile robotics. Due to its inherit characteristics, this method is considered noisy and less robust. On the other hand, due to its popularity, there has been enough research investment to bring about sufficient improvements for the execution and results quality of this technique (e.g. (Lee) and (Heller) can be referred to).
2. *INS methods*: which are based on inertial sensors, accelerometers and detectors for electromagnetic field and gravity. They are also based on integration on movement elements, therefore, may eventually lead to error accumulation especially if drift prune sensors are used. Due to vast field of applications, these methods also enjoyed quite enough completeness, thanks to the developed powerful mathematical tools. (Roos, 2002) is for example offering one of these complementary enhancements.
3. *Vision and visual odometery*: utilizing a single camera, stereo vision or even omni directional imaging, this solution can potentially be useful in giving more information than only working as the localization routine. This solution can considerably become more computationally expensive, especially if employed in a dynamic environment or expected to deal with relatively-non-stationary features. It is however, considered a popular and effective research theme lately, and is enhanced significantly by getting backup support from signal processing techniques, genetic algorithms, as well as evolutionary and learning algorithms.

4. *Ranging*: employing a distance measurement media that can be either laser, infrared, acoustic or radio signals. Ranging can be done using different techniques; recording signal's Time of Flight, Time Difference of Arrival or Round Trip Time of Flight of the beamed signal, as well as its Angle of Arrival. This class is under main interest to be fused properly for increasing efficiency and accuracy of the traditional methods.

There are some less common approaches which indirectly can still be categorized in the classes above. Itemized reference to them is as the following.

Doppler sensors can measure velocity of the moving objects. Principally speaking, a sinusoidal signal is emitted from a moving platform and the echo is sensed at a receiver. These sensors can use ultra sound signal as carrier or radio waves, either. Related to the wavelength, resolution and accuracy may differ; a more resolution is achieved if smaller wavelength (in other words higher frequency) is operational. Here again, this technique works based on integrating velocity vectors over time.

Electromagnetic trackers can determine objects' locations and orientations with a high accuracy and resolution (typically around 1mm in coordinates and 0.2° in orientation). Not only they are expensive methods, but also electromagnetic trackers have a short range (a few meters) and are very sensitive to presence of metallic objects. These limitations only make them proper for some fancy applications such as body tracking computer games, animation studios and so on.

Optical trackers are very robust and typically can achieve high levels of accuracy and resolution. However, they are most useful in well-constrained environments, and tend to be expensive and mechanically complex. Example of this class of positioning devices are head tracker system (Wang et. al, 1990).

Proper fusion of any of the introduced techniques above, can give higher precision in localization but at the same time makes the positioning routine computationally more expensive. For example (Choi & Oh, 2005) combines sonar and vision, (Carpin & Birk, 2004) fuses odometery, INS and ranging, (Fox et. al, 2001) mixes dead reckoning with vision, and there can be found plenty of other practical cases. Besides, each particular method can not be generalized for all applications and might fail under some special circumstances. For instance, using vision or laser rangefinders, should be planned based on having a rough perception about the working environment beforehand (specially if system is working in a dynamic work space. If placed out and not moving, the strategy will differ, eg. in (Bastani et. al, 2005)). Solutions like SLAM which overcome the last weakness, need to detect some reliable features from the work space in order to build the positioning structure (i.e. an evolving representation called as the map) based on them. This category has a high complexity and price of calculation. In this vain, recent technique such as *pose-graph* put significant effort on improvements (Pfingsthhorn & Birk, 2008). Concerning again about the environmental perception; INS solutions and accelerometers, may fail if the work space is electromagnetically noisy. Integrating acceleration and using gravity specifications in addition to the dynamics of the system, will cause a potentially large accumulative error in positioning within a long term use, if applied alone.

3. Ranging Technologies and Wireless Media

The aim here is to refer to ranging techniques based on wireless technology in order to provide some network modelling and indeed realization of the positions of each node in the network. These nodes being mobile robots, form a dynamic (or in short time windows static) topology.
Different network topologies require different positioning system solutions. Their differences come from physical layer specifications, their media access control layer characteristics, some capabilities that their particular network infrastructure provides, or on the other hand, limitations that are imposed by the network structure. We first very roughly categorizes an overview of the positioning solutions including variety of the global positioning systems, those applied on cellular and GSM networks, wireless LANs, and eventually ad hoc sensor networks.

3.1 Wireless Media

Two communicating nodes compose the simplest network where there is only one established link available. Network size can grow by increasing the number of nodes. Based on the completeness of the topology, each node may participate in establishing various number of links. This property is used to define *degree* of a node in the network. Link properties are relatively dependent on the media providing the communication. Bandwidth, signal to noise ratio (SNR), environmental propagation model, transmission power, are some of such various properties. Figure 1 summarizes most commonly available wireless communication media which currently are utilized for network positioning, commercially or in the research fields. With respect to the platform, each solution may provide global or local coordinates which are eventually bidirectionally transformable. Various wireless platforms – based on their inherent specifications and capabilities, may be used such that they fit the environmental conditions and satisfy the localization requirements concerning their accessibility, reliability, maximum achievable resolution and the desired accuracy.

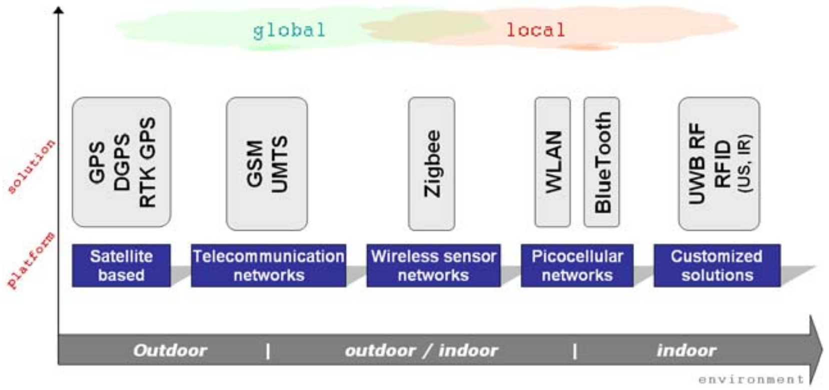


Fig. 1. Sweeping the environment from outdoor to indoor, this figure shows how different wireless solutions use their respective platforms in order to provide positioning. They all indeed use some ranging technique for the positioning purpose, no matter time of flight or received signal strength. Depending on the physical size of the cluster, they provide local or global positions. Anyhow these two are bidirectional transformable.

Obviously, effectiveness and efficiency of a large scale outdoor positioning system is rather different than a small scale isolated indoor one. What basically differs is the environment which they may fit better for, as well as accuracy requirements which they afford to fulfil. On the x-axis of the diagram in figure 1, moving from outdoor towards indoor environment, introduced platforms become less suitable specially due to the attenuation that indoor environmental conditions apply to the signal. This is while from right to left of the x-axis in the same diagram, platforms and solutions have the potential to be customized for outdoor area as well. The only concern is to cover a large enough area outdoor, by pre-installing the infrastructure.
The challenge is dealing with accurate indoor positioning where maximum attenuation is distorting the signal and most of the daily, surveillance and robotics applications are utilized. In this vein, we refer to the WLAN class and then for providing enhancement and more competitive accuracy, will turn to wireless sensor networks.

3.2 Wireless Local Area Networks

Very low price and their common accessibility have motivated development of wireless LAN-based indoor positioning systems such as Bluetooth and Wi-Fi. (Salazar, 2004) comprehensively compares typical WLAN systems in terms of markets, architectures, usage, mobility, capacities, and industrial concerns. WLAN-based indoor positioning solutions mostly depend on signal strength utilization. Anyhow they have either a client-based or client-assisted design.

Client-based system design: Location estimation is usually performed by RF signal strength characteristics, which works much like pattern matching in cellular location systems. Because signal strength measurement is part of the normal operating mode of wireless equipment, as in Wi-Fi systems, no other hardware infrastructure is required. A basic design utilizes two phases. First, in the offline phase, the system is calibrated and a model is constructed based on received signal strength (RSS) at a finite number of locations within the targeted area. Second, during an online operation in the target area, mobile units report the signal strengths received from each access point (AP) and the system determines the best match between online observations and the offline model. The best matching point is then reported as the estimated position.

Client-assisted system design: To ease burden of system management (provisioning, security, deployment, and maintenance), many enterprises prefer client-assisted and infrastructure-based deployments in which simple sniffers monitor client's activity and measure the signal strength of transmissions received from them (Krishnan, 2004). In client-assisted location system design, client terminals, access points, and sniffers, collaborate to locate a client in the WLAN. Sniffers operate in passive scanning mode and sense transmissions on all channels or on predetermined ones. They listen to communication from mobile terminals and record time-stamped information. The sniffers then put together estimations based on a priory model. A client's received signal strength at each sniffer is compared to this model using nearest neighbour searching to estimate the clients location (Ganu, 2004). In terms of system deployment, sniffers can either be co-located with APs, or, be located at other positions and function just like the Location Management Unit in a cellular-based location system.

3.3 Ad hoc Wireless Sensor Networks

Sensor networks vary significantly from traditional cellular networks or similars. Here, nodes are assumed to be small, inexpensive, homogeneous, cooperative, and autonomous. Autonomous nodes in a wireless sensor network (WSN) are equipped with sensing (optional), computation, and communication capabilities. The key idea is to construct an ad hoc network using such wireless nodes whereas nodes' locations can be realized. Even in a pure networking perspective, location-tagged information can be extremely useful for achieving some certain optimization purposes. For example (Kritzler, 2006) can be referred to which proposes a number of location-aware protocols for ad hoc routing and networking.
It is especially difficult to estimate nodes' positions in ad hoc networks without a common clock as well as in absolutely unsynchronized networks. Most of the localization methods in the sensor networks are based on RF signals' properties. However, there are other approaches utilizing Ultra Sound or Infra Red light instead. These last two, have certain disadvantages. They are not omnidirectional in broadcasting and their reception, and occlusion if does not completely block the communication, at least distorts the signal significantly. Due to price flexibilities, US methods are still popular for research applications while providing a high accuracy for in virtu small scale models.
Not completely inclusive in the same category however there are partially similar techniques which use RFID tags and readers, as well as those WSNs that work based on RF UWB communication, all have proven higher potentials for indoor positioning. An UWB signal is a series of very short baseband pulses with time durations of only a few nanoseconds that exist on all frequencies simultaneously, resembling a blast of electrical noise (Fontanaand, 2002). The fine time resolution of UWB signals makes them promising for use in high-resolution ranging. In this category, time of flight is considered rather than the received signal strength. It provides much less unambiguity but in contrast can be distorted by multipath fading. A generalized maximum-likelihood detector for multipaths in UWB propagation measurement is described in (Lee, 2002). What all these techniques are suffering from is needing a centralized processing scheme as well as a highly accurate and synchronized common clock base. Some approaches are however tackling the problem and do not concern time variant functions. Instead, using for example RFID tags and short-range readers, enables them to provide some proximity information and gives a rough position of the tag within a block accuracy/resolution (e.g. the work by (Fontanaand, 2002) with very short range readers for a laboratory environment localization). The key feature which has to be still highlighted in this category is the overall cost of implementation.

4. Infra Structure Principles

In the field of positioning by means of radio signals, there are various measurement techniques that are used to determine position of a node. In a short re-notation they are divided into three groups:

- Distance Measurements: ToF, TDoA, RSS
- Angle Measurements: AoA
- Fingerprinting: RSS patterns (radio maps)

Distance and angle measurement methods are the mostly used metrics for outdoor location systems. Distance measurements use the path loss model and ToF measurements to determine a location. Angle Measurements are based on knowing the angle of incidence of the received signal. However, this class requires directional antennas and antenna arrays to measure the angle of incidence. That makes this option not very viable for a node with high-mobility. For smaller scale applications, this method can be utilized by means of ESPAR (Electronically Steerable Parasitic Array Radiator) antennas. Such an antenna steers autonomously its beam toward the arrival direction of desired radio waves and steers the nulls of the beam toward the undesired interfering waves (Ging, 2005). The versatile beam forming capabilities of the ESPAR antenna allows to reduce multipath fading and makes accurate reading for direction of arrival. There are not too much of applicable experiences for indoor mobile robotics, (Shimizu, 2005) for example, applies it on search and rescue robotics for urban indoor environment.

Distance and Angle measurements work only with direct line-of-sight signals from the transmitter to the receiver, indeed being widely practical for outdoor environments. For indoors, channel consists of multipath components, and the mobile station is probably surrounded by scattering objects. For these techniques to work, a mobile node has to see at least three signal sources, necessarily required for triangulation.

Distance measurement techniques in the simplest case, will end up to multilateration in order to locate position of a desired point, no matter being a transmitter or a receiver. Collaborative multilateration (also referred to as N-hop multilateration) consists of a set of mechanisms that enables nodes to find several hops away from beacon nodes to collaborate with each other and potentially estimate their locations within desired accuracy limits. Collaborative multilateration is presented in two edges of computation models, *centralized* and *distributed*. These can be used in a wide variety of network setups from fully centralized where all the computation takes place at a base station, locally centralized (i.e., computation takes place at a set of cluster heads) to fully distributed where computation takes place at every node.

5. RSS Techniques

A popular set of approaches tries to employ the received signal strength (RSS) as distance estimate between a wireless sender and a receiver. If the physical signal itself can not be measured, packet loss can be used to estimate RSS (Royer, 1999). But the relation between the RF signal strength and spatial distances is very complex as real world environments do not only consist of free space. They also include various objects that cause absorption, reflection, diffraction, and scattering of the RF signals (Rappaport, 1996). The transmitted signal therefore most often reaches the receiver by more than one path, resulting in a phenomenon known as multi path fading (Neskovic et. al, 2000). The quality of these techniques is hence very limited; the spatial resolution is restricted to about a meter and there are tremendous error rates (Xang et. al. 2005). They are hence mainly suited for Context-Awareness with room or block resolution (Yin et. al, 2005). Due to the principle problems with RSS, there are attempts to develop dedicated hardware for localizing objects over medium range.

5.1 Properties

The indoor environment poses additional challenges compared to the outdoor environment. The channel is complex; having various multipath, scattering and diffraction effects that make estimating the signal strength highly probabilistic. Traditional methods working based on AoA and ToF principles can not be used with this technology, additionally special hardware is required to get a time synchronized network out of the available standard access points. Therefore the only applicable method is the received signal strength technique. RSS is the simplest RF measurement technique as its values are easily accessible through a WLAN interface. It is more preferred than the Signal to Noise ratio (SNR) to be used for the radio signature, because it is more location dependant (Bahl & Padmanabhan, 2000). Noise can vary considerably from location to location and may heavily depend on the external factors, while this is not the case for the received signal strength. Since the RSS values still significantly fluctuate over time for a given location, they can be considered as a random variable, and hence, are described in a statistical fashion in order to estimate the distribution parameters. The fundamentals of the RSS techniques come from the Frii's Transmission Equation. The reason that Friis' formula does not work satisfactorily for indoor propagation is that communicating points may suffer from nonl Line of Sight condition (nLoS). Strength decay of the received signal not only comes from path loss, but also shadowing and fading distort the received signal quality. They both depend on the environmental features, barriers and occlusion. Short scale fading due to multipath, adds random high frequency terms with large amplitude (Rappaport, 1996). This issue is more effective indoors. Still because of less complexity that the hardware design and implementation phases need, the RSS solution has been in the field of interest amongst most of the localization researches.

5.2 Techniques

Apart from the statistical or probabilistic representation of signals, there are essentially two categories of RSS based techniques for positioning using WLAN: Trilateration and Location Fingerprinting. The prerequisite of the trilateration method is using a signal propagation model to convert RSS measurement to a transmitter-receiver (T-R) separate distance estimate. Utilizing the general empirical model can only obtain a very inaccurate distance of T-R, therefore a more accurate and correction-enforced model is required. Before a positioning system can estimate the location, a fingerprint database (also referred to as a radio map) constructed. In other words, a "Location Fingerprinting" localization system consists of two phases, the offline (training) and the online (localization) phase. During the online phase a site survey is performed by measuring the RSS values from multiple APs. The floor is divided into predefined grid points. The RSS values are measured at desired locations on the grid. Multiple measurements are taken and the average values are stored in a database. The location fingerprint refers to the vector of RSS values from each access point and their corresponding location on the grid. A reference carpet refers to the collection of fingerprints and their associated locations for a given area. The drawback here is the extensive training values that have to be predetermined, separately for each particular indoor environment.

Understanding the statistical properties of the location fingerprint (aka. RSS vector) is important for the design of a positioning system due to several reasons. It can provide insights into how many APs are needed to uniquely identify a location (Kaemarungsi &

Krishnamurthy, 2004) with a given accuracy and precision, and, whether preprocessing of the RSS measurements can improve such accuracy. Within the same perspective, (Li et. al, 2005) has proposed a hybrid method compose of two stages.
Area localization has been seen as a more viable alternative compared to point based localization mainly due to the fact that they can better describe localization uncertainty (Elnahraway et. al, 2004). They can easily trade accuracy for precision. Accuracy is the likelihood of the object being within the returned area, while precision is the size of the returned area. Point based algorithms on the other hand have difficulty in describing this behaviour. It was found that although area based approaches better described the localization uncertainty, their absolute performance is similar to point based approaches (Elnahraway et. al, 2004).

6. Physical and Logical Topology of WSN

Field of networked robotics is envisioned to be one of the key research areas in robotics recently (Bekey et. al, -). This research field, flourishing in the last decade or so, is gaining additional momentum thanks to the advent of cheap, easy to deploy and low power nodes to build sensor networks. Richness of different sensors, and their prompt availability open new scenarios to solve some long standing problems.
In this scenario, plenty of wireless sensors are deployed to the environment, in order to build up a network. This network first of all aims for holding a full connectivity which in most of applications can be represented by a graph. Plenty of theories and properties of graphs can merge to the scenario and open new doors for improvements (Gotsman & Koren, 2005). However full network connectivity has its own advantages, local connectivities in order to extract connected clusters also are shown to be sufficient under some circumstances (Chan et. al, 2005).
Assume a scenario that builds a mobile wireless (-sensor) network with possibly physical dynamic topology. If there are enough relative distances information available, it is not quite complicated to use some multilateration technique for finding a position, but it is conditioned on the availability of enough anchors to be used as references (Khan et. al, 2006). In this class, many solutions assume the availability of detectable landmarks at known positions in order to implement, for example, Kalman based localization methods (Leonard & Durrant, 1991) and (Pillonetto & Carpin, 2007), while some other approaches are developed based on anchor-free combinations (Pryantha et. al, 2003). Here, we explicitly solve the following layout problem: Given a set of mobile robots (simply named nodes) which are equipped with wireless transceivers and a mechanism by which each node can estimate its distance to a few nearby ones (or even through the whole network), the goal is to determine coordinates of every node via local or global communication.
In general, radio communication constraints are a set of geometry rules to bound position estimates. Such constraints are a combination of radial and angular limitations. Latter aspects are out of the interests of this research. Ranging information is adequately sufficient in for these achievements. It goes to the fact that if all internode wireless links are established and their associative lengths are known, their graph representation is indeed uniquely realized. When only one subset of distances is known, more sophisticated techniques must be used. In contrast, when multiple solutions exist, the main phenomenon observed is that of foldovers, where entire pieces of the graph fold over on top of others,

without violating any of the partial distance constraints. A main challenge is to generate a solution which is fold-free. Occasionally, final result may suffer from translation, rotation and reflection degrees of freedom, but either of these are not important, because can be resolved by assigning some known coordinates to any three arbitrary nodes.
Problem of reconstructing a geometric graph given its edges' lengths, has received some attention in discrete as well as computational geometry communities, whereas it is also relevant for molecule construction and protein folding applications, psychology, networking and also mobile robotics. Our main aim here is to roughly show how to extend graph realization analogy with noisy edges, for localizing the nodes with higher likelihood of unambiguous realization.
In an interconnected network of n nodes, if all distances are known exactly, it is only possible to determine the relative coordinates of the points, i.e., to calculate the point coordinates in some arbitrary system. To place the points in an absolute coordinate system, it is necessary to know the positions of at least $D+1$ points, where D indicates the dimension of the space. Consider that a set of k anchor points are having a known position in the 2D space. A set of n-k nodes with unknown positions can possibly be spatially localized based on the known positions of those anchors. If $k=0$, the positioning scenario is called *relative* or *anchor-less* localization.
On the other hand, an incremental approach can deal with a large size network where nodes are having a limited range of communication and indeed ranging. This solution, assigns coordinates to a small core of nodes, and repeatedly, updates coordinates to more neighbors based on local calculations and then proceeds until the whole network is covered. This solution is totally error prone in the early stages, unless being reinforced by proper filtering methods and statistical approaches, is very likely to break the error upper supremum.

7. Conclusions

In this chapter, after surveying traditional methods for robot localization, we more emphasized on the radio networks and their ranging capabilities including RSS in WLAN networks as well as ToF measurements of WSN topologies. Their positioning accuracy is comparable in practice where the latter provides quite better positioning results. Practical results indicate that the last approach, enforced with some more mathematical representation can be reliably used for variety of mobile robotics positioning applications both indoor and outdoors, relatively independent from size of the mobile robotics network, explicitly explained in (Bastani & Birk, 2009). In other words, having ranging capabilities and specially by radio signals enables the robots to overcome localization problems, less dependently from error accumulation and environmental features.

8. References

Thrun, S (2002). Robotic mapping: A survey, Exploring Artificial Intelligence in the New Millennium, Morgan Kaufmann.
Lee, B; Cai, W,; Turner, J. & Chen, L. (?). Adaptive Dead Reckoning Algorithms for distributed Interactive Simulation, *Journal of SIMULATION*, Vol. 1, No 1-2.
Heller, A. K. & Taylor, S. (?). Using Determinism to Improve the Accuracy of Dead Reckoning Algorithms. Australian National University, CANBERRA.

Roos, T. (2002). A probabilistic approach to wlan user location. *Intl Journal of WirelessInformation Networks*, Vol 9.

Birk, A. (1996). Learning geometric concepts with an evolutionary algorithm. *Procedings of The Fifth Annual Conference on Evolutionary Programming*. The MIT Press, Cambridge.

Salazar, A.E.S. (2004). Positioning bluetooth and Wi-Fi systems, *IEEE Trans. Consumer Electron.*, Vol. 50, No. 1, pp. 151-157.

Krishnan, P., Krishnakumar, A.S., Ju, W.H., Mallows, C. & Ganu, S. (2004) A system for LEASE: System for location estimation assisted by stationary emitters for indoor RF wireless networks, *Proceedings of IEEE Infocom*, pp. 1001-1011. Hong Kong.

Ganu, ., Krishnakumar, A.S. & Krishnan, P. (2004). Infrastructure-based location estimation in WLAN, *Proceedings of IEEE Wireless Communications and Networking Conf. (WCNC 2004)*, pp. 465-470.

Kritzler, M.; Lewejohann, L.; Krger, A.; Raubal, M. & Sachser, N. (2006). An RFID-based Tracking System for Laboratory Mice in a Semi-Natural Environment, *Proceedings In PTA2006 Workshop*, Pervasive Technology Applied Real-World Experiences with RFID and Sensor Networks.

Fontanaand, R. & Gunderson, S. (2002). Ultra wideband precision asset location system, *Proc. IEEE Conf. Ultra Wideband Systems and Technologies*, pp. 147-150.

Lee, J.Y. (2002). Ultra-wideband ranging in dense multipath environments, Ph.D. dissertation, Dept. Elect. Eng., Univ. Southern California.

Royer, E.M. & Toh, C.K. (1999). A review of current routing protocols for ad-hoc mobile wireless networks. *IEEE Personal Communications Magazine*, pp. 46-55.

Rappaport, T.S. (1996). Wireless Comm. Principles and Practice. *IEEE Press /Prentice Hall*.

Neskovic, A.; Nescovic, N. & Paunovic, G. (2000). Modern approaches in modelling of mobile radio systems propagation environment. *IEEE Communications Surveys*.

Xiang, Z.; Zhang, H.; Huang, J.; Song, S.& Almeroth, K. (2005). A hidden environment model for constructing indoor radio maps. *Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM 2005)*. pp. 395-400.

Yin, J.; Yang, Q.& Ni, L. (2005). Adaptive temporal radio maps for indoor location estimation. *3rd IEEE international conf. on Pervasive Computing and Communications, (PerCom 2005)*. pp. 85-94.

Bahl, P & Padmanabhan, V.N. (2000). RADAR: An in-building, RF based user location and tracking system, *Proceedings of IEEE Infocom 2000*, Vol. 2, pp. 775-784. Tel-Aviv, Israel.

Rappaport, T. (1996). Wireless Communications Principle and Practice, *Prentice Hall*.

Kaemarungsi, K. & Krishnamurthy, P. (2004). Modeling of Indoor Positioning Systems Based on Location Fingerprinting. *Proceedings of IEEE INFOCOM 2004*.

Li, B.; Dempster, A.; Rizos, C. & Barnes, J. (2005). Hybrid Method for Localization Using WLAN. School of Surveying and Spatial Information System, University of New South Wales, Sydney.

Elnahraway, E.; Li, X. & Martin, R. P., (2004). The limits of localization using RSS. *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pp. 283-284, New York, USA.

Sklar. B.(1997). Rayleigh fading channels in mobile digital communication systems, *IEEE Communications Magazine*, Vol. 35, pp. 90-100.

Thrun, S. (2000). An Online Mapping Algorithm for Teams of Mobile Robots, CMU-CS-00-167.

Bekey, G.; Ambrose, R.;V. Kumar ; A. Sanderson ; B. Wilcox, & Y. Zheng. (?). International assessment of research and development in robotics.

Gotsman C. & Koren, Y. (2005). Distributed graph layout for sensor networks, *Journal of graph algorithms and applications*, Vol.9, No.1.

Chan, H.; Luk M. & Perrig, A. (2005). Using Clustering Information for Sensor Network Localization, *LNCS, Springer, Distributed Computing in Sensor Systems*, Vol. 3560, pp. 109-125.

Khan, H.M. ; Olariu, S. & Eltoweissy, M., (2006). Efficient single-anchor localization in sensor networks, Dependability and Security in Sensor Networks and Systems, DSSNS.

Bastani, H. & Birk, A. (2009). Precise Realtime Localization by RF Transceiver ToF Measurements, *14th International Conference on Advanced Robotics (ICAR)*, Germany.

Leonard, J. & Durrant-Whyte, H. (1991). Mobile robot localization by tracking geometric beacons, *IEEE Transaction on Robotics and Automation*, Vol.7, No. 3, pp.:376-382.

Pillonetto, G. & Carpin, S. (2007). Multirobot localization with unknown variance parameters using iterated Kalman filtering, *IEEE/RSJ International Conference on Intelligent Robots and Systems*. pp.: 1709-1714.

Priyantha, N.; Balakrishnan, H.; Demanie, E. & Teller, S. (2003). Anchor-free distributed localization in sensor networks, *Proceedings of SenSys 03*.

Horn, K. (1987). Closed-form solution of absolute orientation using unit quaternion, *Journal of Optical Society*. USA, Vol.4, pp.: 629-638.

Wang, J.; Chi, V. & Fuchs, C. (1990). A Real-time Optical 3D Tracker for Head-mounted Display Systems, Computer Graphics, *ACM SIGGRAPH*, Vol. 24, No.2, pp.205-215.

Choi, Y. & Oh, S. (2005). Visual Sonar Based Localization Using Particle Attraction and Scattering, *Proceedings of the IEEE International Conference on Mechatronics and Automation*, Canada.

Carpin, S. & Birk, A. (2004). Stochastic map merging in rescue environments, In Robocup 2004. *Springer*.

Fox, D.; Thrun, S.; Burgard, W. & Dellaert, F. (2001). Particle filters for mobile robot localization. Sequential Monte Carlo Methods in Practice, *Springer Verlag,* pp. 499-516.

Bastani, H. & Mirmohammad Sadeghi, H.; Shariatmadari, A.; Azarnasab, E. & Davarpanah Jazi, M. (2005). Design and Implementation of a Full Autonomous Mine Detecting Robot, 5th IFAC on Intelligent Autonomous Vehicles - IAV04, *Elsevier, ISBN 008 044237 4*.

Pfingsthorn, M. & Birk, A. (2008). Efficiently Communicating Map Updates with the Pose Graph, *International Conference on Intelligent Robots and Systems (IROS)*, 2008.

Qing, H. (2005) A compact ESPAR antenna with planer parasitic elements on a dielectric cylinder, *IEICE trans.communication.* Vol.E88-B.

Shimizu, M. (2005). Tiny Seekers, Robocup Rescue - Robot League Team, *TDP in Robocup 2005 proceeding* (CD format), Japan.

Basic Extended Kalman Filter – Simultaneous Localisation and Mapping

[1] Oduetse Matsebe, [2] Molaletsa Namoshe and [3] Nkgatho Tlale
[1,2,3] *Council for Scientific and Industrial Research, Mechatronics and Micro Manufacturing Pretoria, South Africa*

1. Introduction

Most real systems are non-linear. Extended Kalman Filter (EKF) uses non-linear models of both the process and observation models while the Kalman Filter (KF) uses linear models. EKF is a good way to learn about Simultaneous Localisation and Mapping (SLAM). Much of the literature concentrates on advanced SLAM methods which stems from EKF or uses probabilistic techniques. This makes it difficult for new researchers to understand the basics of this exciting area of research.
SLAM asks if it is possible for a robot, starting with no prior information, to move through its environment and build a consistent map of the entire environment. Additionally, the vehicle must be able to use the map to navigate and hence plan and control its trajectory during the mapping process. The applications of a robot capable of navigating, with no prior map, are diverse indeed. Domains in which 'man in the loop' systems are impractical or difficult such as sub-sea surveys and disaster zones are obvious candidates. Beyond these, the sheer increase in autonomy that would result from reliable, robust navigation in large dynamic environments is simply enormous (Newman 2006). SLAM has been implemented in a number of different domains from indoor robots to outdoor, underwater, and airborne systems. In many applications the environment is unknown. A priori maps are usually costly to obtain, inaccurate, incomplete, and become outdated. It also means that the robot's operation is limited to a particular environment (Neira 2008).
This goal of the chapter is to provide an opportunity for researchers who are new to, or interested in, this exciting area with the basics, background information, major issues, and the state-of-the-art as well as future challenges in SLAM with a bent towards EKF-SLAM. It will also be helpful in realizing what methods are being employed and what sensors are being used. It presents the 2 – Dimensional (2D) feature based EKF-SLAM technique used for generating robot pose estimates together with positions of features in the robot's operating environment, it also highlights some of the basics for successful EKF – SLAM implementation: (1) Process and observation models, these are the underlying models required, (2) EKF-SLAM Steps, the three-stage recursive EKF-SLAM process comprising prediction, observation and update, (3) Feature Extraction and Environment modelling, a

process of extracting well defined entities or landmarks (features) which are recognisable and can be repeatedly detected to aid navigation, (4) Data Association, this consists of determining the origin of each measurement, in terms of map features, (5) Multi – Robot – EKF – SLAM, the two types of multi robot systems are described: Collaborative and Cooperative multi robot systems with more emphasis on the Cooperative SLAM Scheme.

2. Basic Structure of EKF - SLAM

The EKF-SLAM process consists of a recursive, three-stage procedure comprising prediction, observation and update steps. The EKF estimates the pose of the robot made up of the position (x_r, y_r) and orientation ψ_r, together with the estimates of the positions of the N environmental features $\boldsymbol{x}_{f,i}$ where $i = 1....N$, using observations from a sensor onboard the robot (Williams et al. 2001). We will constrain ourselves to using the simplest feature model possible; a point feature such that the coordinates of the i^{th} feature in the global reference frame are given by:

$$\boldsymbol{x}_{f,i} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{1}$$

SLAM considers that all landmarks are stationary, hence the state transition model for the i^{th} feature is given by:

$$\boldsymbol{x}_{f,i}(k) = \boldsymbol{x}_{f,i}(k-1) = \boldsymbol{x}_{f,i} \tag{2}$$

It is important to note that the evolution model for features does have any uncertainty since the features are considered static.

2.1 Process Model

Implementation of EKF - SLAM requires that the underlying state and measurement models be developed. This section describes the process models necessary for this purpose.

2.1.1 Kinematic Model

Modeling of the kinematic states involves the study of the geometrical aspects of motion. The motion of a robot through the environment can be modeled through the following discrete time non-linear model:

$$\boldsymbol{X}_r(k) = \boldsymbol{f}(\boldsymbol{X}_r(k-1), \boldsymbol{u}(k), k) \tag{3}$$

Thus, $\boldsymbol{X}_r(k)$ is the state of the robot at time k, $\boldsymbol{u}(k)$ is the robot control input at time k. $\boldsymbol{f}(,,)$ is a function that relates the robot state at time $k-1$, known control inputs to the robot state at time k.

$$\boldsymbol{u}(k) = \begin{bmatrix} u_1 \\ . \\ . \\ u_N \end{bmatrix} \tag{4}$$

Equation (4) above is a little unrealistic, we need to model uncertainty. One popular way to model uncertainty is to insert noise terms into the control input $\boldsymbol{u}(k)$ such that:

$$\boldsymbol{u}(k) = \boldsymbol{u}_n(k) + \boldsymbol{\gamma}_u(k) \tag{5}$$

Thus $\boldsymbol{u}_n(k)$ is a nominal control input and $\boldsymbol{\gamma}_u(k)$ is a vector of control noise which is assumed to be temporally uncorrelated, zero mean and Gaussian with standard deviation σ.

$$\gamma_u(k) = \begin{bmatrix} \sigma_1 \\ . \\ . \\ \sigma_N \end{bmatrix} \tag{6}$$

The strength (covariance) of the control noise is denoted $\boldsymbol{Q}_u$, and is given by:

$$\boldsymbol{Q}_u = diag\left(\sigma_1^2 \quad . \quad . \quad \sigma_N^2\right) \tag{7}$$

The complete discrete time non-linear kinematic model can now be expressed in general form as:

$$\boldsymbol{X}_r(k) = f(\boldsymbol{X}_r(k-1), \boldsymbol{u}_n(k) + \boldsymbol{\gamma}_u(k)) \tag{8}$$

2.1.2 Using Dead-Reckoned Odometry Measurements

Sometimes a navigation system will be given a dead reckoned odometry position as input without recourse to the control signals that were involved. The dead reckoned positions can be converted into a control input for use in the core navigation system. It would be a bad idea to simply use a dead-reckoned odometry estimate as a direct measurement of state in a Kalman Filter (Newman 2006).

Given a sequence $\boldsymbol{x}_o(1), \boldsymbol{x}_o(2), \boldsymbol{x}_o(3)...\boldsymbol{x}_o(k)$ of dead reckoned positions, we need to figure out a way in which these positions could be used to form a control input into a navigation system. This is given by:

$$\boldsymbol{u}_o(k) = \Theta \boldsymbol{x}_o(k-1) \oplus \boldsymbol{x}_o(k) \tag{9}$$

This is equivalent to going back along $\boldsymbol{x}_o(k-1)$ and forward along $\boldsymbol{x}_o(k)$. This gives a small control vector $\boldsymbol{u}_o(k)$ derived from two successive dead reckoned poses. Equation (9) substracts out the common dead-reckoned gross error (Newman 2006). The plant model for a robot using a dead reckoned position as a control input is thus given by:

$$\boldsymbol{X}_r(k) = \boldsymbol{f}(\boldsymbol{X}_r(k-1), \boldsymbol{u}(k)) \tag{10}$$

$$\boldsymbol{X}_r(k) = \boldsymbol{X}_r(k-1) \oplus \boldsymbol{u}_o(k) \tag{11}$$

Θ and $\oplus$ are composition transformations which allows us to express robot pose described in one coordinate frame, in another alternative coordinate frame. These composition transformations are given below:

$$\boldsymbol{x}_1 \oplus \boldsymbol{x}_2 = \begin{bmatrix} x_1 + x_2 \cos\theta_1 - y_2 \sin\theta_1 \\ y_1 + x_2 \sin\theta_1 + y_2 \cos\theta_1 \\ \theta_1 + \theta_2 \end{bmatrix} \tag{12}$$

$$\Theta \boldsymbol{x}_1 = \begin{bmatrix} -x_1 \cos\theta_1 - y_1 \sin\theta_1 \\ x_1 \sin\theta_1 - y_1 \cos\theta_1 \\ -\theta_1 \end{bmatrix} \tag{13}$$

2.2 Measurement Model

This section describes a sensor model used together with the above process models for the implementation of EKF - SLAM. Assume that the robot is equipped with an external sensor capable of measuring the range and bearing to static features in the environment. The measurement model is thus given by:

$$\boldsymbol{z}(k) = \boldsymbol{h}(\boldsymbol{X}_r(k), x_i, y_i) + \boldsymbol{\gamma}_h(k) = \begin{bmatrix} r_i(k) \\ \theta_i(k) \end{bmatrix} \tag{14}$$

$$r_i = \sqrt{\left(x_i - x_r\right)^2 + \left(y_i - y_r\right)^2} \tag{15}$$

$$\theta_i = \tan^{-1}\left[\frac{y_i - y_r}{x_i - x_r}\right] - \psi_r \tag{16}$$

(x_i, y_i) are the coordinates of the i^{th} feature in the environment. $\boldsymbol{X}_r(k)$ is the (x,y) position of the robot at time k. $\boldsymbol{\gamma}_h(k)$ is the sensor noise assumed to be temporally uncorrelated, zero mean and Gaussian with standard deviation σ. $r_i(k)$ and $\theta_i(k)$ are the range and bearing respectively to the i^{th} feature in the environment relative to the vehicle pose.

$$\boldsymbol{\gamma}_h(k) = \begin{bmatrix} \sigma_r \\ \sigma_\theta \end{bmatrix} \tag{17}$$

The strength (covariance) of the observation noise is denoted $\boldsymbol{R}$.

$$\boldsymbol{R} = diag\left(\sigma_r^2 \quad \sigma_\theta^2\right) \tag{18}$$

2.3 EKF SLAM Steps

This section presents the three-stage recursive EKF-SLAM process comprising prediction, observation and update steps. Figure 1 below summarises the EKF - SLAM process described here.

2.3.1 Map Initialisation

The selection of a base reference B to initialise the stochastic map at time step 0 is important. One way is to select as base reference the robot's position at step 0. The advantage in choosing this base reference is that it permits initialising the map with perfect knowledge of the base location (Castellanos et al. 2006).

$$\boldsymbol{X}_0^B = \boldsymbol{X}_r^B = 0 \tag{19}$$

$$\boldsymbol{P}_0^B = \boldsymbol{P}_r^B = 0 \tag{20}$$

This avoids future states of the vehicle's uncertainty reaching values below its initial settings, since negative values make no sense. If at any time there is a need to compute the vehicle location or the map feature with respect to any other reference, the appropriate transformations can be applied. At any time, the map can also be transformed to use a feature as base reference, again using the appropriate transformations (Castellanos et al. 2006).

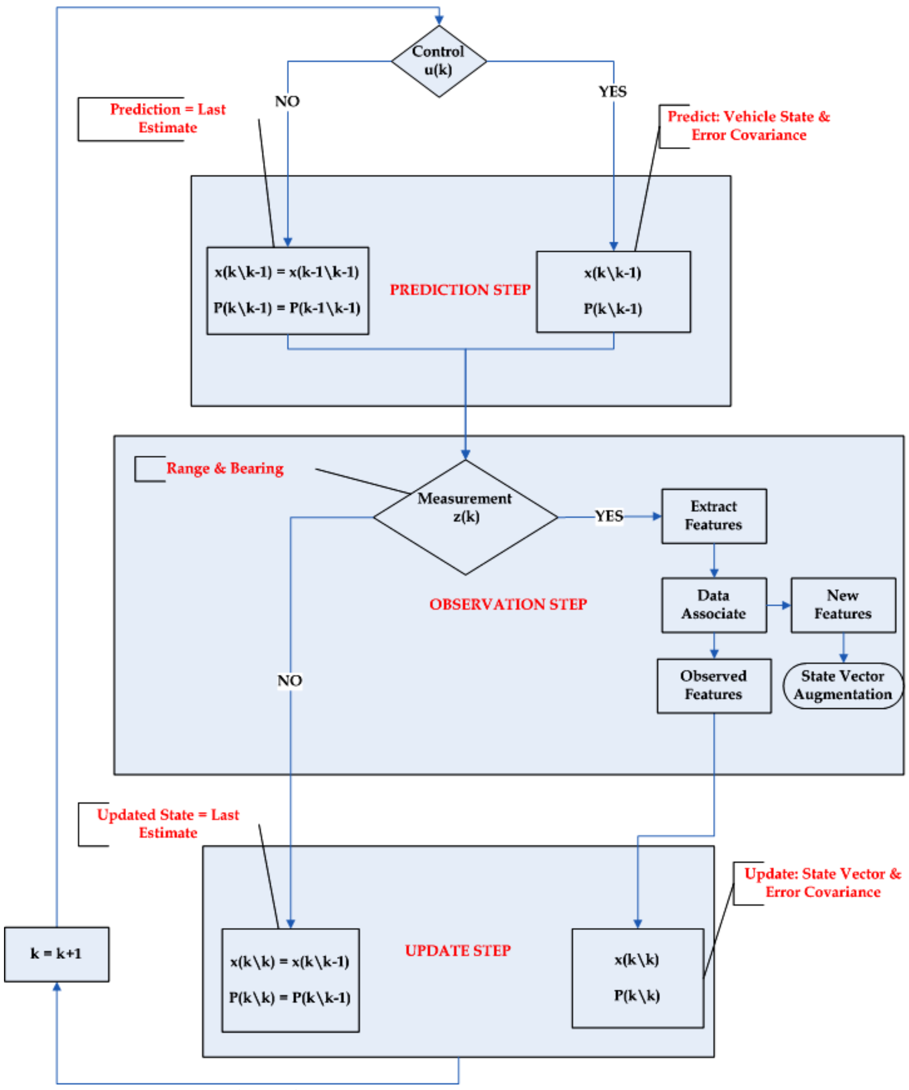


Fig. 1. Basic EKF-SLAM Flow chart

2.3.2 Prediction

(a) Prediction based on kinematic model

The prediction stage is achieved by passing the last estimate through the non-linear model of motion of the robot to compute the pose at instant k based on a control input $\boldsymbol{u}(k)$ and using the information up to instant $k-1$ (Williams et al. 2001). The predicted robot state $\boldsymbol{X}_r$ is thus given by:

$$\boldsymbol{X}_r(k\,|\,k-1) = \boldsymbol{f}(\boldsymbol{X}_r(k-1\,|\,k-1), \boldsymbol{u}(k)) \tag{21}$$

Now we need to propagate the state error covariance. The covariance of the robot state, $\boldsymbol{P}_r(k \backslash k-1)$ is computed using the gradient, $\boldsymbol{F}_x(k)$ of the state propagation equation (8) with respect to the robot pose, the control noise covariance, $\boldsymbol{Q}_u$ and, the Jacobian, $\boldsymbol{J}_u$ of the state propagation equation (8) with respect to the control input $\boldsymbol{u}(k)$.

$$\boldsymbol{P}_r(k\,|\,k-1) = \boldsymbol{F}_x(k)\boldsymbol{P}_r(k-1\,|\,k-1)\boldsymbol{F}_x^T(k) + \boldsymbol{J}_u(k)\boldsymbol{Q}_u(k)\boldsymbol{J}_u^T(k) \tag{22}$$

$$\boldsymbol{F}_x(k) = \frac{\partial f}{\partial X_r} \tag{23}$$

$$\boldsymbol{J}_u(k) = \frac{\partial f}{\partial u} \tag{24}$$

(b) Prediction using Dead-Reckoned Odometry Measurements as inputs

The prediction stage is achieved by a composition transformation of the last estimate with a small control vector calculated from two successive dead reckoned poses.

$$\boldsymbol{X}_r(k\,|\,k-1) = \boldsymbol{X}_r(k-1\,|\,k-1) \oplus \boldsymbol{u}_o(k) \tag{25}$$

The state error covariance of the robot state, $\boldsymbol{P}_r(k \backslash k-1)$ is computed as follows:

$$\boldsymbol{P}_r(k\,|\,k-1) = \boldsymbol{J}_1(\boldsymbol{X}_r,\boldsymbol{u}_o)\boldsymbol{P}_r(k-1\,|\,k-1)\boldsymbol{J}_1(\boldsymbol{X}_r,\boldsymbol{u}_o)^T + \boldsymbol{J}_2(\boldsymbol{X}_r,\boldsymbol{u}_o)\boldsymbol{U}_O(k)\boldsymbol{J}_1(\boldsymbol{X}_r,\boldsymbol{u}_o)^T \tag{26}$$

$\boldsymbol{J}_1(\boldsymbol{X}_r,\boldsymbol{u}_o)$ is the Jacobian of equation (11) with respect to the robot pose, $\boldsymbol{X}_r$ and $\boldsymbol{J}_2(\boldsymbol{X}_r,\boldsymbol{u}_o)$ is the Jacobian of equation (11) with respect to the control input, $\boldsymbol{u}_o$.

Based on equations (12), the above Jacobians are calculated as follows:

$$\boldsymbol{J}_1(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{\partial(\boldsymbol{x}_1 \oplus \boldsymbol{x}_2)}{\partial \boldsymbol{x}_1} \tag{27}$$

$$\boldsymbol{J}_1(\boldsymbol{x}_1, \boldsymbol{x}_2) = \begin{bmatrix} 1 & 0 & -x_2 \sin\theta_1 - y_2 \cos\theta_1 \\ 0 & 1 & -x_2 \cos\theta_1 - y_2 \sin\theta_1 \\ 0 & 0 & 1 \end{bmatrix} \tag{28}$$

$$\boldsymbol{J}_2(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{\partial(\boldsymbol{x}_1 \oplus \boldsymbol{x}_2)}{\partial \boldsymbol{x}_2} \tag{29}$$

$$J_2(\boldsymbol{x}_1, \boldsymbol{x}_2) = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{30}$$

2.3.3 Observation

Assume that at a certain time k an onboard sensor makes measurements (range and bearing) to m features in the environment. This can be represented as:

$$\boldsymbol{z}_m(k) = [\boldsymbol{z}_1 \quad . \quad . \quad \boldsymbol{z}_m] \tag{31}$$

2.3.4 Update

The update step need not happen at every iteration of the filter. If at a given time step no observations are available then the best estimate at time k is simply the prediction $\boldsymbol{X}(k \mid k-1)$. If an observation is made of an existing feature in the map, the state estimate can now be updated using the optimal gain matrix $\boldsymbol{W}(k)$. This gain matrix provides a weighted sum of the prediction and observation. It is computed using the innovation covariance $\boldsymbol{S}(k)$, the state error covariance $\boldsymbol{P}(k \mid k-1)$ and the gradient of the observation model (equation 14), $\boldsymbol{H}(k)$.

$$\boldsymbol{W}(k) = \boldsymbol{P}(k \mid k-1)\boldsymbol{H}(k)\boldsymbol{S}^{-1}(k), \tag{32}$$

where $\boldsymbol{S}(k)$ is given by:

$$\boldsymbol{S}(k) = \boldsymbol{H}(k)\boldsymbol{P}(k \mid k-1)\boldsymbol{H}^T(k) + \boldsymbol{R}(k) \tag{33}$$

$\boldsymbol{R}(k)$ is the observation covariance.

This information is then used to compute the state update $\boldsymbol{X}(k \mid k)$ as well as the updated state error covariance $\boldsymbol{P}(k \mid k)$.

$$\boldsymbol{X}(k \mid k) = \boldsymbol{X}(k \mid k-1) + \boldsymbol{W}(k)\nu(k) \tag{34}$$

$$\boldsymbol{P}(k \mid k) = \boldsymbol{P}(k \mid k-1) - \boldsymbol{W}(k)\boldsymbol{S}(k)\boldsymbol{W}(k)^T \tag{35}$$

The innovation, $\boldsymbol{\nu}(k)$ is the discrepancy between the actual observation, $\boldsymbol{z}(k)$ and the predicted observation, $\boldsymbol{z}(k \mid k-1)$.

$$\boldsymbol{\nu}(k) = \boldsymbol{z}(k) - \boldsymbol{z}(k \mid k-1), \tag{36}$$

where $\boldsymbol{z}(k \mid k-1)$ is given as:

$$\boldsymbol{z}(k \mid k-1) = \boldsymbol{h}\left(\boldsymbol{X}_r(k \mid k-1), \boldsymbol{x}_i, \boldsymbol{y}_i\right) \tag{37}$$

$X_r(k \mid k-1)$ is the predicted pose of the robot and (x_i, y_i) is the position of the observed map feature.

2.4 Incorporating new features

Under SLAM the system detects new features at the beginning of the mission and when exploring new areas. Once these features become reliable and stable they are incorporated into the map becoming part of the state vector. A feature initialisation function $\boldsymbol{y}$ is used for this purpose. It takes the old state vector, $\boldsymbol{X}(k \mid k)$ and the observation to the new feature, $\boldsymbol{z}(k)$ as arguments and returns a new, longer state vector with the new feature at its end (Newman 2006).

$$\boldsymbol{X}(k \mid k)^* = \boldsymbol{y}\left[\boldsymbol{X}(k \mid k), \quad \boldsymbol{z}(k)\right] \tag{38}$$

$$\boldsymbol{X}(k \mid k)^* = \begin{bmatrix} \boldsymbol{X}(k \mid k) \\ x_r + r\cos(\theta + \psi_r) \\ y_r + r\sin(\theta + \psi_r) \end{bmatrix} \tag{39}$$

Where the coordinates of the new feature are given by the function $\boldsymbol{g}$:

$$\boldsymbol{g} = \begin{bmatrix} x_r + r\cos(\theta + \psi_r) \\ y_r + r\sin(\theta + \psi_r) \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} \tag{40}$$

r and θ are the range and bearing to the new feature respectively. (x_r, y_r) and ψ_r are the estimated position and orientation of the robot at time k .
The augmented state vector containing both the state of the vehicle and the state of all feature locations is denoted:

$$\boldsymbol{X}(k \mid k)^* = [\boldsymbol{X}_r^T(k) \quad \boldsymbol{x}_{f,1}^T \quad . \quad . \quad \boldsymbol{x}_{f,N}^T] \tag{41}$$

We also need to transform the covariance matrix $\boldsymbol{P}$ when adding a new feature. The gradient for the new feature transformation is used for this purpose:

$$\boldsymbol{g} = \begin{bmatrix} x_r + r\cos(\theta + \psi_r) \\ y_r + r\sin(\theta + \psi_r) \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} \tag{42}$$

The complete augmented state covariance matrix is then given by:

$$\boldsymbol{P}(k \mid k)^* = \boldsymbol{Y}_{x,z} \begin{bmatrix} \boldsymbol{P}(k \mid k) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R} \end{bmatrix} \boldsymbol{Y}_{x,z}^T, \tag{43}$$

where $\boldsymbol{Y}_{x.z}$ is given by:

$$\boldsymbol{Y}_{x,z} = \begin{bmatrix} \boldsymbol{I}_{nxn} & \boldsymbol{0}_{nx2} \\ [\boldsymbol{G}_{x_r} \quad zeros(nstates - n)] & \boldsymbol{G}_z \end{bmatrix} \tag{44}$$

where $nstates$ and n are the lengths of the state and robot state vectors respectively.

$$\boldsymbol{G}_{X_r} = \frac{\partial g}{\partial X_r} \tag{45}$$

$$\boldsymbol{G}_{X_r} = \begin{bmatrix} \frac{\partial g_1}{\partial x_r} & \frac{\partial g_1}{\partial y_r} & \frac{\partial g_1}{\partial \psi_r} \\ \frac{\partial g_2}{\partial x_r} & \frac{\partial g_2}{\partial y_r} & \frac{\partial g_2}{\partial \psi_r} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -r\sin(\theta + \psi_r) \\ 0 & 1 & r\cos(\theta + \psi_r) \end{bmatrix} \tag{46}$$

$$\boldsymbol{G}_z = {}^{\partial g}\!/_{\partial z} \tag{47}$$

$$\boldsymbol{G}_z = \begin{bmatrix} \frac{\partial g_1}{\partial r} & \frac{\partial g_1}{\partial \theta} \\ \frac{\partial g_2}{\partial r} & \frac{\partial g_2}{\partial \theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta+\psi_r) & -r\sin(\theta+\psi_r) \\ \sin(\theta+\psi_r) & r\cos(\theta+\psi_r) \end{bmatrix} \tag{48}$$

2.5 Structure of state covariance matrix

The covariance matrix has some structure to it. It can be partitioned into map $\boldsymbol{P}_{mm}$ and the robot $\boldsymbol{P}_{rr}$ covariance blocks.

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{P}_{rr} & \boldsymbol{P}_{rm} \\ \boldsymbol{P}_{mr} & \boldsymbol{P}_{mm} \end{bmatrix} \tag{49}$$

Off diagonals $\boldsymbol{P}_{rm}$ and $\boldsymbol{P}_{mr}$ blocks are the correlations between map and robot since they are interrelated. From the moment of initialisation, the feature position is a function of the robot position hence errors in the robot position will also appear as errors in the feature position. Every observation of a feature affects the estimate of every other feature in the map (Newman 2006).

3. Consistency of EKF SLAM

SLAM is a non-linear problem hence it is necessary to check if it is consistent or not. This can be done by analysing the errors. The filter is said to be unbiased if the Expectation of the actual state estimation error, $\tilde{\boldsymbol{X}}(k)$ satisfies the following equation:

$$E\left[\tilde{\boldsymbol{X}}(k)\right] = 0 \tag{50}$$

$$E\left[\left(\tilde{\boldsymbol{X}}(k)\right)\left(\tilde{\boldsymbol{X}}(k)\right)^T\right] \leq \boldsymbol{P}(k \mid k-1) \tag{51}$$

, where the actual state estimation error is given by:

$$\tilde{\boldsymbol{X}}(k) = \boldsymbol{X}(k) - \boldsymbol{X}(k \mid k-1) \tag{52}$$

$\boldsymbol{P}(k \mid k-1)$ is the state error covariance. Equation (51) means that the actual mean square error matches the state covariances. When the ground truth solution for the state variables is available, a chi-squared test can be applied on the normalised estimation error squared to check for filter consistency.

$$\left(\tilde{\boldsymbol{X}}(k)\right)^T \left(\boldsymbol{P}(k \mid k-1)\right)^{-1} \left(\tilde{\boldsymbol{X}}(k)\right) \leq \chi^2_{d,1-\alpha} \tag{53}$$

$d = \dim(x(k))$ and $1-\alpha$ is the desired confidence level. In most cases ground truth is not available, and consistency of the estimation is checked using only measurements that satisfy the innovation test:

$$\boldsymbol{v}_{ij}^T(k)\boldsymbol{S}_{ij}^{-1}\boldsymbol{v}_{ij}(k) < \chi^2_{d,1-\alpha} \tag{54}$$

Since the innovation term depends on the data association hypothesis, this process becomes critical in maintaining a consistent estimation of the environment map (Castellanos et al 2006).

4. Feature Extraction and Environment modelling

This is a process by which sensor data is processed to obtain well defined entities (features) which are recognisable and can be repeatedly detected. In feature based navigation methods, features must be different from the rest of the environment representation (**discrimination).** To be able to re-observe features, they must be **invariant** in scale, dimension or orientation, and they must also have a well defined **uncertainty model.** In structured domains such as indoor, features are usually modelled as geometric primitives such as points, lines and surfaces. Contrary to indoor domains, natural environments cannot be simply modelled as geometric primitives since they do not conform to any simple geometric model. A more general feature description is necessary in this regard. To aid feature recognition in these environments, more general shapes or blobs can be used and characteristics such as size, centre of mass, area, perimeter, aspects such as colour, texture, intensity and other pertinent information descriptors like mean, and variance can be extracted (Ribas 2005).

5. Data Association

In practice, features have similar properties which make them good landmarks but often make them difficult to distinguish one from the other. When this happen the problem of data association has to be addressed. Assume that at time k , an onboard sensor obtains a set of measurements $\boldsymbol{z}_i(k)$ of m environment features $\boldsymbol{E}_i(i=1,...,m)$. Data Association consists of determining the origin of each measurement, in terms of map features $F_j, j=1,...,n.$ The results is a hypothesis:

$$\boldsymbol{H}_k = \begin{bmatrix} j_1 & j_2 & j_3 \ldots\ldots j_m \end{bmatrix} \tag{55}$$

, matching each measurement $\boldsymbol{z}_i(k)$ with its corresponding map feature. $F_{ji}(j_i = 0)$ indicates that the measurement $\boldsymbol{z}_i(k)$ does not come from any feature in the map (Castellanos et al. 2006). Figure 2 below summarises the data association process described here. Several techniques have been proposed to address this issue and more information on some these techniques can be found in (Castellanos et al 2006) and (Cooper 2005).
Of interest in this chapter is the simple data association problem of finding the correspondence of each measurement to a map feature. Hence the Individual Compatibility Nearest Neighbour Method will be described.

5.1 Individual Compatibility (IC)

The IC considers individual compatibility between a measurement and map feature (Castellanos et al. 2006). This idea is based on a prediction of the measurement that we would expect each map feature to generate, and a measure of the discrepancy between a predicted measurement and an actual measurement made by the sensor. The predicted measurement is then given by:

$$\boldsymbol{z}_j(k \mid k-1) = \boldsymbol{h}(\boldsymbol{X}_r(k \mid k-1), x_j, y_j) \tag{56}$$

The discrepancy between the observation $\boldsymbol{z}_i(k)$ and the predicted measurement $\boldsymbol{z}_j(k \mid k-1)$ is given by the innovation term $\boldsymbol{v}_{ij}(k)$:

$$\boldsymbol{v}_{ij}(k) = \boldsymbol{z}_i(k) - \boldsymbol{z}_j(k \mid k-1) \tag{57}$$

The covariance of the innovation term is then given as:

$$\boldsymbol{S}_{ij}(k) = \boldsymbol{H}(k)\boldsymbol{P}(k \mid k-1)\boldsymbol{H}^T(k) + \boldsymbol{R}(k) \tag{58}$$

$\boldsymbol{H}(k)$ is made up of $\boldsymbol{H}_r$, which is the Jacobian of the observation model with respect to the robot states and $\boldsymbol{H}_{Fj}$, the gradient Jacobian of the observation model with respect to the observed map feature.

$$\boldsymbol{H}(k) = \begin{bmatrix} \boldsymbol{H}_r & 0\,0 & 0\,0 & \boldsymbol{H}_{Fj} & 0\,0 \end{bmatrix} \tag{59}$$

Zeros in equation (59) above represents un-observed map features.

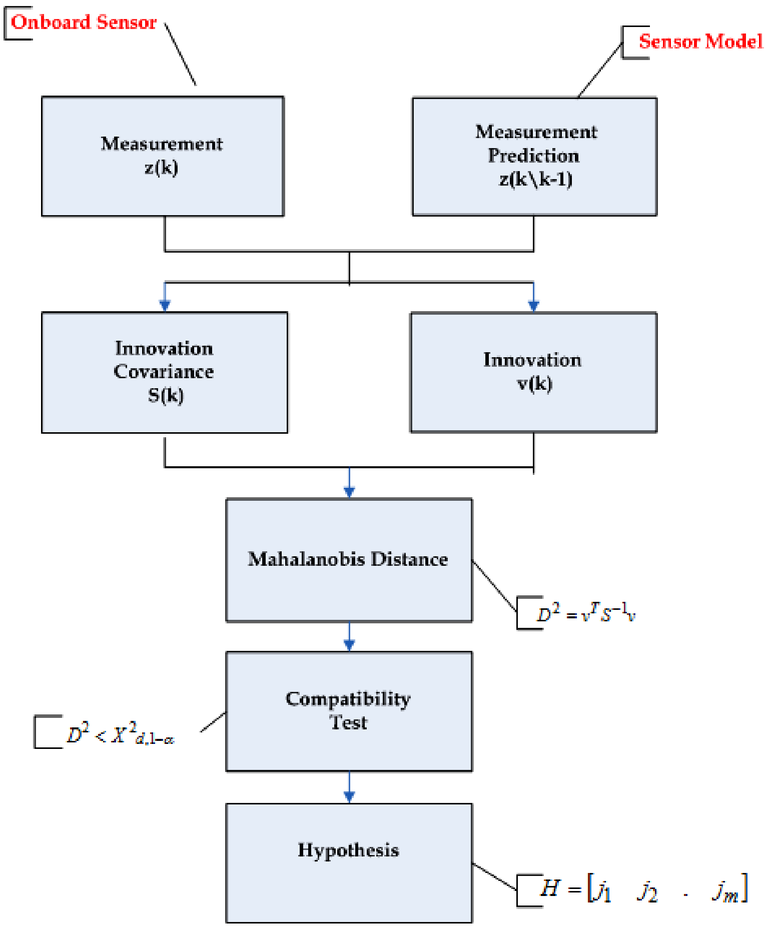


Fig. 2. Data Association Flow chart

To deduce a correspondence between a measurement and a map feature, Mahalanobis distance is used to determine compatibility, and it is given by:

$$D_{ij}^2(k) = \boldsymbol{v}_{ij}^T(k)\boldsymbol{S}_{ij}^{-1}(k)\boldsymbol{v}_{ij}(k) \tag{60}$$

The measurement and a map feature can be considered compatible if the Mahalanobis distance satisfies:

$$D_{ij}^2(k) < \chi_{d,1-\alpha}^2 \tag{61}$$

Where $d = \dim(v_{ij})$ and $1-\alpha$ is the desired level of confidence usually taken to be 95%. The result of this exercise is a subset of map features that are compatible with a particular measurement. This is the basis of a popular data association algorithm termed Individual Compatibility Nearest Neighbour. Of the map features that satisfies IC, ICNN chooses one with the smallest Mahalanobis distance (Castellanos et al 2006).

6. Commonly used SLAM Sensors

The most popular sensor choice in indoor and outdoor applications is a laser scanner even though it is costly. Its popularity stems from the fact that it provides high quality dense data with a good angular precision. Cameras are also commonly used to obtain visual information (e.g colour, shape or texture) from the environment. Acoustic sensors are considered to be the cheapest choice but less reliable to perform SLAM even in highly structured environments. This is because sonars produce measurements with poor angular resolution and the problem of specular reflections. If used, then one must somehow deal with these limitations. The situation is different in underwater domains. Due to the attenuation and dispersion of light in water, laser based sensors and cameras become impractical, though cameras can still be used in applications where the vehicle navigates in clear water or very near to the seafloor. Due to excellent propagation of sound in water, acoustic sensors remain the best choice in the underwater domain (Ribas 2008).

7. Multi – Robot EKF – SLAM

In order for a multi-robot team to coordinate while navigating autonomously within an area, all robots must be able to determine their positions as well as map the navigation map with respect to a base frame of reference. Ideally, each robot would have direct access to measurements of its absolute position such as using GPS, but this is not possible indoor or in the vicinity of tall structures. Therefore, utilising multi robot systems becomes attractive as robots can operate individually but use information from each other to correct their estimates (Mourikis, Roumeliotis 2005).

7.1 Collaboration Vs Cooperation

There are two types of multi robot systems: collaborative and cooperative multi robot system. Collaborative case is when robots working as a team in real-time, continuously update each others state estimates with the latest sensor information. While cooperative kind is when robots share information via an external computer to find the group solution based on available communicated information (Andersson, L. 2009).
Of interest in this chapter is the Cooperative SLAM Scheme. Assuming we have two robots capable of individually performing SLAM and robot–robot relative measurements as shown in figure 3. In the picture, SLAM results and as well as possible robot-robot relative measurements results are fed into the Cooperative strategy module to calculate possible route for each member. The module utilises the current global map information to derive controls for the system. That is, determining the team manoeuvres providing optimal reward in terms of exploration gain. Figure 3 below shows a Cooperative SLAM Scheme which is an estimation and control closed loop problem similar to the structure discussed in (Andrade-Cetto,J. et al. 2005).

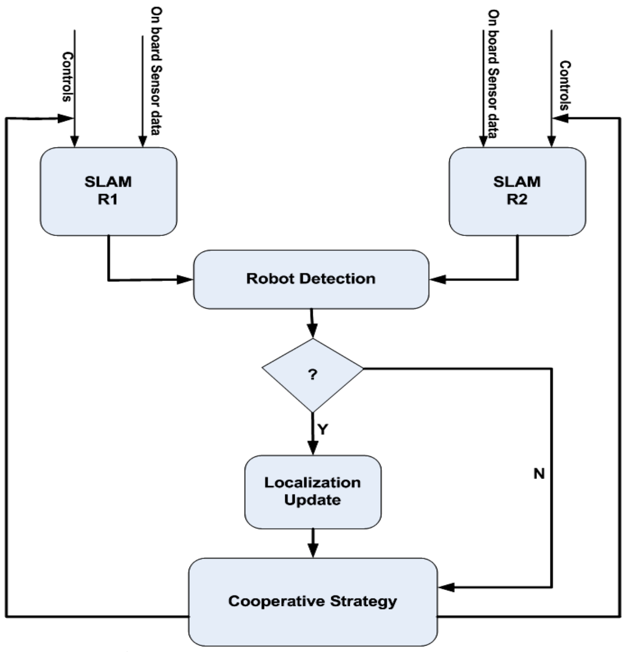


Fig. 3. Cooperative SLAM scheme

7.2 Map Initialisation

As in single robot SLAM, Multi robot SLAM system requires some common reference frame between the robots. We then make assumptions that, (1) initially the global map is empty, (2) robot one R_i starts at known pose.

$$\hat{\boldsymbol{x}}_i^B = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T \tag{62}$$

While the second robot R_j is placed in front of the first and is detectable by R_i.

7.3 State Augmentation

The first assumption states that the map is initially empty; therefore the known robots and objects need to be added to the map before updating. Using stochastic representation as discussed in (Smith, Self & Cheesman 1986), priori information can be inserted in the estimated state vector and corresponding covariance matrix as shown in equation (63 & 64) below.

$$\hat{\boldsymbol{x}}^B = \begin{bmatrix} \hat{\boldsymbol{x}}_i^B \\ \hline \hat{\boldsymbol{x}}_j^B \end{bmatrix} \tag{63}$$

$$\boldsymbol{P}^B = \begin{bmatrix} P(\boldsymbol{x}_i) & P(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ P(\boldsymbol{x}_j, \boldsymbol{x}_i) & P(\boldsymbol{x}_j) \end{bmatrix} \tag{64}$$

Noting that all information inserted into the state vector $\hat{\boldsymbol{x}}^B$ needs to be represented in the global reference frame B. From the second assumption, when a robot is observed by R_i for the first time, the system state vector needs to be augmented by the pose of the observed robot with respect to frame of the observing robot. The observed pose estimate is then transformed from the observer frame of reference to global frame through the following equations:

$$\boldsymbol{g}^B = \hat{\boldsymbol{x}}_j^B = \hat{\boldsymbol{x}}_i^{\,B} + R\left(\hat{z}_j^i\right) \tag{65}$$

$$R\left(\hat{z}_j^i\right) = \begin{bmatrix} r\cos(\theta_j^i + \psi_i) \\ r\sin(\theta_j^i + \psi_i) \end{bmatrix}, \tag{66}$$

where $R\left(\hat{z}_j^i\right)$ is a nonlinear transform function giving the spatial perception by the observer robot. The corresponding observation covariance matrix $\boldsymbol{R}$ is represented in the observer's reference frame and also needs to be transformed into global frame as shown in equation (67), i.e. $\boldsymbol{G}_z\boldsymbol{R}\boldsymbol{G}_z$. The aforementioned measurement covariance matrix $\boldsymbol{R}$ differs with sensor type, but in this chapter, we model the laser range finder type of sensor, providing range and bearing. Where range and bearing are respectively given by r and θ_j^i from equation (66). And ψ_i is the orientation of the observer robot.

A correct stochastic map needs the independences and interdependences to be maintained through at the mapping process. Since no absolute object (robots or landmarks) locations are given prior, the estimations of objects positioning are correlated and strongly influenced by the uncertainty in the robot(s) locations. Equation (64) is a covariance matrix of the system, where the leading diagonal elements are the variances of the state variables for R_i and R_j respectively. These are evaluated similar to the derivation in (Martinelli, Pont & Siegwart 2005), that is:

$$\boldsymbol{P}_{(k-1|k-1)}(\boldsymbol{x}_i) = \nabla\boldsymbol{f}_{k-1}^{\;i}\boldsymbol{P}_{(k-1|k-1)}(\boldsymbol{x}_i)\nabla\boldsymbol{f}_{k-1}^{\;iT} + \boldsymbol{G}_z\boldsymbol{R}\boldsymbol{G}_z\,, \tag{67}$$

where $\nabla\boldsymbol{f}_{k-1}^{\;i}$ is the Jacobian of equation (65) with respect to the observer robot state, derived as;

$$\nabla \boldsymbol{f}_{k-1}{}^{i} = \frac{\partial \boldsymbol{g}^{B}}{\partial x_{i}{}^{B}} = \begin{bmatrix} 1 & 0 & -\rho \sin(\theta_{j}^{i} + \psi_{i}) \\ 0 & 1 & \rho \cos(\theta_{j}^{i} + \psi_{i}) \\ 0 & 0 & 1 \end{bmatrix} \tag{68}$$

G_z is also the Jacobian of equation (65) with respect to the observation. And it is calculated as:

$$\boldsymbol{G}_{z} = \frac{\partial \boldsymbol{g}^{B}}{\partial z_{j}{}^{i}} = \begin{bmatrix} \cos(\theta_{j}^{i} + \psi_{i}) & -r \sin(\theta_{j}^{i} + \psi_{i}) \\ \sin(\theta_{j}^{i} + \psi_{i}) & r \cos(\theta_{j}^{i} + \psi_{i}) \end{bmatrix} \tag{69}$$

The off diagonal elements are computed as:

$$\boldsymbol{P}(x_j, x_i) = \nabla \boldsymbol{f}_{k-1}^{j} \boldsymbol{P}(x_j, x_i) \nabla \boldsymbol{f}_{k-1}^{i}{}^{T}, \tag{70}$$

where

$$\boldsymbol{P}(x_j, x_i) = \boldsymbol{P}(x_j, x_i)^{T} \tag{71}$$

The results of equations (67),(70) and (71) are then substituted back into equation (64) giving the stochastic map estimates at the initial poses.
The discrete conditional subscript used in equation (67) and the reminder of the paper is for readability purposes therefore, $\boldsymbol{P}_{k-1|k-1}$ implies $\boldsymbol{P}(k-1 \mid k-1)$.

7.4 Robot Re-observation

If a robot already in the map is re-observed, the system state vector is not augmented but it is updated. Thus, if we assume that robot R$_i$ located at pose $\boldsymbol{x}_i^B$ makes an observation $\boldsymbol{z}_j^i$ of another robot at pose $\boldsymbol{x}_j^B$ then observation function is given as:

$$\boldsymbol{z}_j^i = \boldsymbol{h}(\boldsymbol{x}_j, \boldsymbol{x}_i) + \boldsymbol{w}_{k-1} \tag{72}$$

Assuming $\boldsymbol{w}_{k-1}$ is zero mean and of Gaussian distribution form, with covariance $\boldsymbol{R}$, calculated as:

$$\boldsymbol{R} = \begin{bmatrix} \Re_i & 0_{3\times3} \\ 0_{3\times3} & \Re_j \end{bmatrix} \tag{73}$$

Suppose the two robot are equipped with the same sensor, then $\Re_i$ and $\Re_i$ are the observation covariances for the corresponding robot. The term $\boldsymbol{h}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ in equation (72) is a nonlinear function that relates the output of the sensor to the states. The function is made up of relative distance and relative bearing (R$_i$ observes R$_j$), given as:

$$\boldsymbol{h}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \begin{bmatrix} \sqrt{(x_j - x_i)^2 \quad + \quad (y_j - y_i)^2} \\ a\tan\left(\dfrac{-\sin\psi_i(x_j - x_i) + \cos\psi_i(y_j - y_i)}{\cos\psi_i(x_j - x_i) + \sin\psi_i(y_j - y_i)} \right) \end{bmatrix} \tag{74}$$

Its jacobian is calculated as follows:

$$\nabla\boldsymbol{h} = \begin{bmatrix} -\dfrac{x_j - x_i}{(x_j - x_i)^2 \ + \ (y_{r2} - y_{r1})^2} & -\dfrac{y_j - y_i}{(x_j - x_i)^2 \ + \ (y_j - y_i)^2} & 0 & \dfrac{x_j - x_i}{(x_j - x_i)^2 \ + \ (y_j - y_i)^2} & \dfrac{y_j - y_i}{(x_j - x_i)^2 \ + \ (y_j - y_i)^2} & 0 \\ \dfrac{y_j - y_i}{(x_j - x_i)^2 \ + \ (y_j - y_i)^2} & -\dfrac{x_j - x_i}{(x_j - x_i)^2 \ + \ (y_j - y_i)^2} & -1 & -\dfrac{y_j - y_i}{(x_j - x_i)^2 \ + \ (y_j - y_i)^2} & \dfrac{x_j - x_i}{(x_j - x_i)^2 \ + \ (y_j - y_i)^2} & 0 \end{bmatrix} \tag{75}$$

Given prior and current measurement information we can update state estimate using Extended Kalman Filter (EKF). The filter update equation is evaluated as:

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}^B_{k|k-1} + \boldsymbol{K}\left[\boldsymbol{z}^j_j - \boldsymbol{h}(\hat{\boldsymbol{x}}^B_j, \hat{\boldsymbol{x}}^B_i) \right] \tag{76}$$

$$\boldsymbol{P}_{k|k} = \boldsymbol{P}_{k|k-1} - \boldsymbol{K}\nabla\boldsymbol{h}\boldsymbol{P}_{k|k-1} \tag{77}$$

$$\boldsymbol{K} = \boldsymbol{P}_{k|k-1}\nabla\boldsymbol{h}^T\left(\nabla\boldsymbol{h}\boldsymbol{P}_{k|k-1}\nabla\boldsymbol{h}^T + \boldsymbol{R} \right)^{-1} \tag{78}$$

$\hat{\boldsymbol{x}}_{k|k}$ implies $\hat{x}(k-1 \mid k-1)$.

8. Conclusions

EKF is a good way to learn about SLAM because of its simplicity whereas probabilistic methods are complex but they handle uncertainty better. This chapter presents some of the basics feature based EKF-SLAM technique used for generating robot pose estimates together with positions of features in the robot's operating environment. It highlights some of the basics for successful EKF – SLAM implementation:, these include: Process and observation models, Basic EKF-SLAM Steps, Feature Extraction and Environment modelling, Data Association, and Multi – Robot – EKF – SLAM with more emphasis on the Cooperative

SLAM Scheme. Some of the main open challenges in SLAM include: SLAM in large environments, Large Scale Visual SLAM, Active and Action based SLAM; development of intelligent guidance strategies to maximise performance of SLAM, Autonomous Navigation and Mapping in Dynamic Environments, and 3D Mapping techniques.

9. References

Andersson, L.(2009). Multi-robot Information Fusion: Considering spatial uncertainty models, Doctoral thesis, Linköping University

Andrade-Cetto J, T. Vidal-Calleja, and A. Sanfeliu.(2005). Multirobot C-SLAM: Simultaneous localization, control, and mapping. In *Proceedings of the IEEE ICRA Workshop on Network Robot Systems*, Barcelona.

Castellanos, J.A.; Neira, J.; Tardos, J.D. (2006). Map Building and SLAM Algorithms, *Autonomous Mobile Robots: Sensing, Control, Decision Making and Applications*, Lewis, F.L. & Ge, S.S. (eds), 1st edn, pp 335-371, CRC, 0-8493-3748-8, New York, USA

Cooper, A.J. (2005). A Comparison of Data Association Techniques for Simultaneous Localisation and Mapping, Masters Thesis, Massachusets Institute of Technology

Martinelli, A., Pont, F. & Siegwart, R. (2005). Mult-irobot localization using relative observations, *Proceedings of the IEEE International Conference on Robotics and Automation*, pp 2808-2813

Mourikis, A.I. & Roumeliotis, S.I. (2005). Performance Bounds for Cooperative Simultaneous Localization and Mapping (C-SLAM), *In Proc. Robotics: Science and Systems Conference*, June 8-11, 2005, pp. 73-80, Cambridge, MA

Neira, J. (2008). The state of the art in the EKF solution to SLAM, *Presentation*

Newman, P. (2006). EKF Based Navigation and SLAM, *SLAM Summer School 2006*

Ribas, D. (2008). Underwater SLAM For Structured Environments Using and Imaging Sonar, *Phd Thesis*, University of Girona

Ribas, D. (2005). Towards Simultaneous Localization & Mapping for an AUV using an Imaging Sonar, *Masters Thesis*, University of Girona

Smith, R., Self, M. & Cheesman, P. (1986), Estimating uncertain spatial relationships in robotics, *Proceedings of the 2nd Annual Conference on Uncertainty in Artificial Intelligence, (UAI-86)*, pp. 435–461, Elsevier Science Publishing Company, Inc., New York, NY

Williams, S.B.; Newman, P.; Rosenblatt, J.; Dissanayake, G. & Whyte, H.D. (2001). Autonomous underwater navigation and control, *Robotica*, vol. 19, no. 5, pp. 481-496

4

Model based Kalman Filter Mobile Robot Self-Localization

Edouard Ivanjko, Andreja Kitanov and Ivan Petrović
University of Zagreb, Faculty of Electrical Engineering and Computing,
Department of Control and Computer Engineering
Croatia

1. Introduction

Mobile robot self-localization is a mandatory task for accomplishing its full autonomy during navigation. Various solutions in robotics community have been developed to solve the self-localization problem. Developed solutions can be categorized into two groups: relative localization (dead-reckoning) and absolute localization. Although very simple and fast, dead-reckoning algorithms tend to accumulate errors in the system as they utilize only information from proprioceptive sensors as odometer readings (e.g. incremental encoders on the robot wheels). Absolute localization methods are based on exteroceptive sensors information. These methods yield a stable locating error but are more complex and costly in terms of computation time. A very popular solution for achieving online localization consists of combining both relative and absolute methods. Relative localization is used with a high sampling rate in order to maintain the robot pose up-to-date, whereas absolute localization is applied periodically with a lower sampling rate to correct relative positioning misalignments Borenstein et al. (1996a).
As regards absolute localization within indoor environments, map-based approaches are common choices. In large majority of cases, it is assumed that a map (model) of the workspace has been established. The environment model can either be pre-stored as architects drawing CAD model or built online simultaneously with localization using sensor data fusion or structure from motion technics. The classical approach to model-based localization consists of matching the local representation of the environment built from sensor information with the global model map and will be used in this chapter also.
So, this chapter presents two approaches to mobile robot self-localization regarding used perceptive sensor combined with an environment model. First approach uses sonar range sensor and second approach uses monocular camera. Environment model in the first approach is an occupancy grid map, and second approach uses a 3D rendered model.
Sonar range sensor is often used in mobile robotics for localization or mapping tasks Lee (1996); Wijk (2001) especially after occupancy grid maps were introduced Moravec & Elfes (1985). Mostly feature maps are used because of their more accurate environment presentation. In such a case when sonar range measurement prediction is done additional steps have to be made. First, appropriate environment feature has to be detected and then corresponding uncertainty that detected feature is correct has to be computed. This uncertainty adds also additional computation step into mostly used Kalman filter framework for non-linear

systems making pose estimation more computationally complex. In this chapter an occupancy grid map is used as the environment model in combination with Extended Kalman filter (EKF) and Unscented Kalman filter (UKF). In this way computation of detected feature uncertainty is avoided and doesn't have to be included in the Kalman filter localization framework with sonar sensor.

General Kalman filter consist of a prediction (time update) and correction (measurement update) step Welch & Bishop (2000). Prediction step makes an update of the estimated state. In this case estimated state is mobile robot pose (combined mobile robot position and orientation). So a kinematic mobile robot model is used for state prediction. In field of mobile robotics such a kinematic model is also known as odometry Borenstein et al. (1996b). It uses measured drive wheel speeds to compute mobile robot pose displacement from a known start pose. It is accurate for short distances because of its aproximate quadratical error growth rate. Such a growth rate arises from the fact that pose error in current pose estimation time step is added to all previous made errors. Fortunately some odometry errors can be taken into account slowing down the error growth rate Borenstein et al. (1996b). This is done by odometry calibration and is also used in work described in this chapter.

Although navigation using vision has been addressed by many researchers, vision is not commonly used on its own but usually in conjunction with other exteroceptive sensors, where multi-sensor fusion techniques are applied, see e.g. Arras et al. (2001) and Li et al. (2002). However, cameras have many advantages as range sensors comparing to sonars and laser rangefinders as they are passive sensors, provide much more data about environment, can distinguish between obstacles based on color etc. A great deal of work has been done on stereo vision, see e.g. Guilherme & Avinash (2002), but for reasons of size and cost monocular vision based navigation has been addressed by a number of researchers, e.g. Aider et al. (2005); Jeon & Kim (1999); Kosaka & Kak (1992); Neira et al. (1997).

When using monocular vision, the localization process is composed of the five following stages Guilherme & Avinash (2002); Kosaka & Kak (1992): 1) image acquisition from current robot pose; 2) image segmentation and feature extraction; 3) model rendering; 4) matching 2D-image and 3D-model features and 5) camera pose computation. It is observed that each stage may be time-consuming due to large amount of data involved. The strategy ultimately adopted for each phase must then be very well-assessed for real-time use. For example, an efficient real-time solution to the feature matching problem is presented in Aider et al. (2005), where interpretation tree search techniques were applied. For mobile robot working environment modelling and rendering professional freeware computer graphics tool Blender *www.blender3d.org* (1995) was used, which is an open source software for 3D modelling, animation, rendering, post-production, interactive creation and playback. It is available for all major operating systems and under the GNU General Public License. The main advantage of that choice is getting powerful 3D modelling tool while being able to optimize the code for user application and use some external well proven computer graphics solutions that Blender incorporates: OpenGL and Python. It also gives external renderer Yafray. Real-time image segmentation for complex and noisy images is achieved by applying Random Window Randomized Hough Transform (RWRHT) introduced in Kälviäinen et al. (1994) which is here used for the first time for robot self-localization to the best of our knowledge. We also implemented and improved robot (camera) pose estimation algorithm developed in Kosaka & Kak (1992).

This chapter is organized as follows. Second section describes used sensors including their mathematical models. Following section describes applied sensor calibration procedures.

Fourth section describes sonar based localization, followed with the fifth section describing monocular vision based localization. Sixth section gives experimental results obtained with a differential drive mobile robot including experiment condition description and comment of localization results. Chapter ends with conclusion.

2. Mobile robot and sensors models

As mentioned in the introduction, used sensors are encoders, sonars and a mono camera. Whence encoders measure mobile robot velocity which is used for odometric pose estimation this section will describe mobile robot kinematic model used for odometric pose estimation. Besides odometry, sonar and camera models will be described.

2.1 Mobile robot kinematics model

As mentioned above a differential drive mobile robot for indoor environments is used in experiments. Its kinematics configuration with denoted relevant variables is presented in Fig. 1. Such a mobile robot configuration has three wheels. Two front wheels are drive wheels with encoder mounted on them and the third wheel is a castor wheel needed for mobile robot stability. Drive wheels can be controlled independently. Kinematic model of differential drive mobile robot is given by the following relations:

$$x(k+1) = x(k) + D(k) \cdot \cos\left(\Theta(k) + \frac{\Delta\Theta(k)}{2}\right), \tag{1}$$

$$y(k+1) = y(k) + D(k) \cdot \sin\left(\Theta(k) + \frac{\Delta\Theta(k)}{2}\right), \tag{2}$$

$$\Theta(k+1) = \Theta(k) + \Delta\Theta(k), \tag{3}$$

$$D(k) = v_t(k) \cdot T, \tag{4}$$

$$\Delta\Theta(k) = \omega(k) \cdot T, \tag{5}$$

$$v_t(k) = \frac{\omega_L(k)R + \omega_R(k)R}{2}, \tag{6}$$

$$\omega(k) = \frac{\omega_R(k)R - \omega_L(k)R}{b}, \tag{7}$$

where are: $x(k)$ and $y(k)$ coordinates of the center of axle [mm]; $D(k)$ travelled distance between time step k and $k+1$ [mm]; $v_t(k)$ mobile robot translation speed [mm/s]; $\omega(k)$ mobile robot rotational speed [°/s]; T sampling time [s]; $\Theta(k)$ angle between the vehicle and x-axis [°]; $\Delta\Theta(k)$ rotation angle between time step k and $k+1$ [°]; $\omega_L(k)$ and $\omega_R(k)$ angular velocities of the left and right wheel, respectively [rad/s]; R radius of the two drive wheels [mm], and b mobile robot axle length [mm]. This general model assumes that both drive wheels have equal radius. Sampling time T was $0.1\,[s]$.

In case of real world mobile robot operations, drive wheel speed measurements are under measurement error influence and some mobile robot parameters values aren't exactly known. Measurement error is mostly a random error with zero mean and can't be compensated. Unknowing exact mobile robot parameters present systematic error and can be compensated

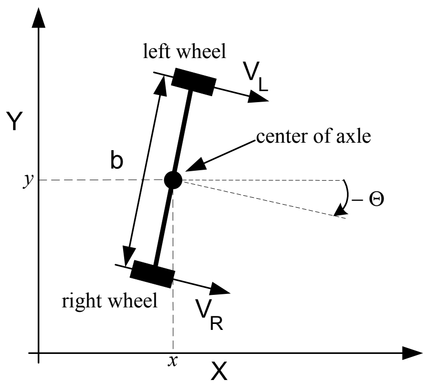


Fig. 1. Differential drive mobile robot kinematics

by means of calibration. Crucial unknown parameters are drive wheel radii and axle length. Wheel radii effect the measurement of the drive wheel circumference speed and axle length measurement affects the mobile robot rotational speed. Equations (6) and (7) can so be rewritten to present mentioned error influence:

$$v_t(k) = \frac{(\omega_L(k)R + \varepsilon_L) + (\omega_R(k)R + \varepsilon_R)}{2}, \tag{8}$$

$$\omega(k) = \frac{(\omega_R(k)R + \varepsilon_R) - (\omega_L(k)R + \varepsilon_L)}{b + \varepsilon_b}, \tag{9}$$

where ε_L, ε_R and ε_b are the error influences on the drive wheel circumference speed measurements and axle length, respectively. It can be noticed here that axle length is also under influence of systematic and random errors. Systematic error obviously comes from unknowing the exact axle length. In this case random error is caused by the effective axle length, which depends on the wheel and surface contact points disposition. Contact points disposition may wary during mobile robot motion due to uneven surfaces and influence of non-symmetric mass disposition on the mobile robot during rotation.

2.2 Sonar model

An interpretation of measured sonar range is given in Fig. 2. It can be seen that in 2D a sonar range measurement can be presented as a part of a circle arc. Size of circle part is defined by the angle of the main sonar lobe and is typical for of the shelf sonar's between 20 and 30 degrees. Therefore, the detected obstacle is somewhere on the arc defined by measured range and main sonar's lobe angle. Background of Fig. 2 shows a grid which is used for creation of occupancy grid map. When a sonar range measurement is interpreted in an occupancy grid framework usually a one to two cells wide area around the measured range is defined as the occupied space. Space between the sonar sensor and measured range is empty space. The sonar is a time of flight sensor, which means it sends a wave (acoustic in this case) and measures the time needed for returning the wave reflected from an obstacle back to the sonar. Generated acoustic wave has its most intensity along its axis, as denoted in Fig. 2, therefore resulting a more accurate distance measurement of obstacles that are inline and perpendicular to the sonar axis. Whence wave intensity decreases with traversed distance, absolute range measurement accuracy also decreases with wave-traversed distance. This is related with requirement of a big range measurement which is a longer open time window to accept the reflected wave and therefore enable more specular reflections and outliers. Specular reflections and outliers present in this case false readings, which decrease the quality of the obtained map. To take this sonar range measurement features into account a stronger emphasis is given to the range measurements closer to the sonar sensor and environment parts closer to the main sonar axis. Mathematically this can be expressed with following equations [3]:

$$\alpha(\Theta) = \begin{cases} 1 - \left(\frac{\Theta}{\Theta_0}\right)^2 & 0 \le \Theta \le \Theta_0 \\ 0 & |\Theta| > \Theta_0 \end{cases}, \tag{10}$$

$$\Delta(\rho) = 1 - \frac{1 + \tanh\left(2\left(\rho - \rho_v\right)\right)}{2}, \tag{11}$$

where $\alpha(\Theta)$ presents angular modulation function i.e., main lobe pattern of the used sonar sensor; Θ angle between sonar axis and currently updated cell $[^\circ]$; Θ_0 is one half of the sonar main lobe angle $[^\circ]$; ρ distance from the sonar sensor and currently updated cell $[m]$; $\Delta(\rho)$ presents radial modulation function and ρ_v presents visibility radius where less emphasis is given to the sonar range measurement further away from visibility radius $[m]$. Parameter Θ_0 value depends from the used sonar sensor and for our Polaroid 6500 sonar sensor it is $12.5\,[^\circ]$. Parameter ρ_v decreases influences of outlier readings and recommended value for an indoor environment is $1.2\,[m]$. Sonar range measurement uncertainty is modeled with angular and radial modulation functions. Most accurate range measurements are so within main sonar axis which is used later in sonar range measurement prediction.

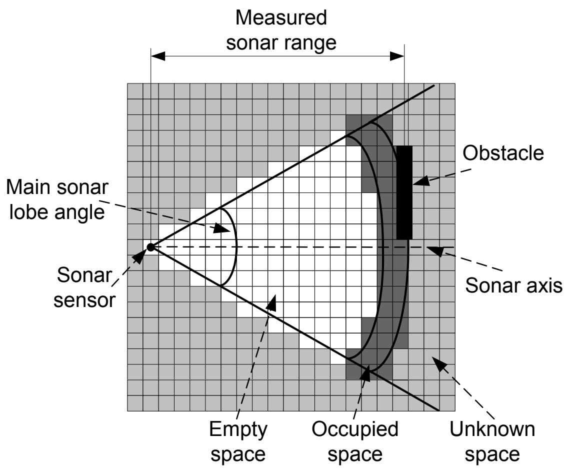


Fig. 2. Interpretation of a sonar range measurement

2.3 Camera model

Generally, a camera has 6 degrees of freedom in three-dimensional space: translations in directions of axes x, y and z, which can be described with translation matrix $T(x, y, z)$, and rotations around them with angles α, β and γ, which can be described with rotation matrices $R_x(\alpha), R_y(\beta)$ and $R_z(\gamma)$. Camera motion in the world coordinate system can be described as the composition of translation and rotation matrices:

$$C = T(x, y, z)\ R_z(\gamma)\ R_y(\beta)\ R_x(\alpha), \tag{12}$$

where

$$\mathbf{R_x(a)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha & -sin\alpha & 0 \\ 0 & sin\alpha & cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{R_y(b)} = \begin{bmatrix} cos\beta & 0 & sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -sin\beta & 0 & cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{R_z(g)} = \begin{bmatrix} cos\gamma & -sin\gamma & 0 & 0 \\ sin\gamma & cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{T(x, y, z)} = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Inverse transformation C^{-1} is equal to extrinsic parameters matrix that is

$$C^{-1}(\alpha, \beta, \gamma, x, y, z) = R_x(-\alpha)\ R_y(-\beta)\ R_z(-\gamma) T(-x, -y, -z). \tag{13}$$

Perspective projection matrix then equals to $P = S\,C^{-1}$ where S is intrinsic parameters matrix determined by off-line camera calibration procedure described in Tsai (1987). The camera is approximated with full perspective pinhole model neglecting image distortion:

$$\left[(x, y)^\top = \left(\frac{\alpha_x X_c}{Z_c} + x_0, \frac{\alpha_y Y_c}{Z_c} + y_0\right)^\top\right], \tag{14}$$

where $\alpha_x = f/s_x$ and $\alpha_y = f/s_y$, s_x and s_y are pixel height and width, respectively, f is camera focal length, (X_c, Y_c, Z_c) is a point in space expressed in the camera coordinate system and $(x_0, y_0)^\top$ are the coordinates of the principal (optical) point in the retinal coordinate system. The matrix notation of (14) is given with:

$$\begin{bmatrix} WX \\ WY \\ W \end{bmatrix} = \underbrace{\begin{bmatrix} \alpha_x & 0 & x_0 & 0 \\ 0 & \alpha_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{S} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}. \tag{15}$$

In our implementation, the mobile robot moves in a plane and camera is fixed to it at the height h, which leaves the camera only 3 degrees of freedom. Therefore, the camera pose is equal to the robot pose $\mathbf{p}$. Having in mind particular camera definition in Blender, the following transformation of the camera coordinate system is necessary $C^{-1}(-\pi/2, 0, \pi + \varphi, p_x, p_y, h)$ in order to achieve the alignment of its optical axes with z, and its x and y axes with the retinal coordinate system. Inverse transformation C^{-1} defines a new homogenous transformation of 3D points from the world coordinate system to the camera coordinate system:

$$C^{-1} = \begin{bmatrix} -cos\varphi & -sin\varphi & 0 & cos\varphi \; p_x + sin\varphi \; p_y \\ 0 & 0 & -1 & h \\ sin\varphi & -cos\varphi & 0 & -sin\varphi \; p_x + cos\varphi \; p_y \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{16}$$

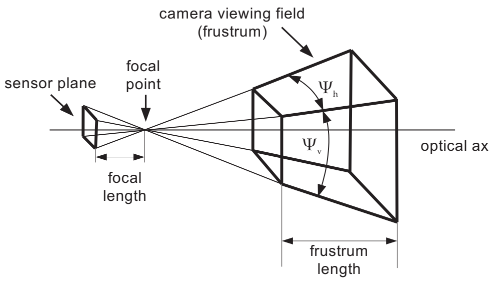


Fig. 3. Visible frustrum geometry for pinhole camera model

Apart from the pinhole model, the full model of the camera should also include information on the camera field of view (frustrum), which is shown in Fig. 3. The frustrum depends on the camera lens and plane size. Nearer and further frustrum planes correspond to camera lens depth field, which is a function of camera space resolution. Frustrum width is defined with angles Ψ_h and Ψ_v, which are the functions of camera plane size.

3. Sensors calibration

Sensor models given in the previous section describe mathematically working principles of sensors used in this article. Models include also influence of real world errors on the sensors measurements. Such influences include system and nonsystem errors. System errors are constant during mobile robot usage so they can be compensated by calibration. Calibration can significantly reduce system error in case of odometry pose estimation. Sonar sensor isn't so influenced by error when an occupancy grid map is used so its calibration is not necessary. This section describes used methods and experiments for odometry and mono-camera calibration. Obtained calibration parameters values are also given.

3.1 Odometry calibration

Using above described error influences, given mobile robot kinematic model can now be augmented so that it can include systematic error influence and correct it. Mostly used augmented mobile robot kinematics model is a three parameters expanded model Borenstein et al. (1996b) where each variable in the kinematic model prone to error influence gets an appropriate calibration parameter. In this case each drive wheel angular speed gets a calibration parameter and third one is attached to the axle length. Using this augmentation kinematics model given with equations (8) and (9) can now be rewritten as:

$$v_t(k) = \frac{(k_1\omega_L(k)R + \varepsilon_{Lr}) + (k_2\omega_R(k)R + \varepsilon_{Rr})}{2}, \tag{17}$$

$$\omega(k) = \frac{(k_2\omega_R(k)R + \varepsilon_{Rr}) - (k_1\omega_L(k)R + \varepsilon_{Lr})}{k_3 b + \varepsilon_{br}}, \tag{18}$$

where ε_{Lr}, ε_{Rr}, and ε_{br} are the respective random errors, k_1 and k_2 calibration parameters that compensate the unacquaintance of the exact drive wheel radius, and k_3 unacquaintance of the exact axle length.
As mentioned above, process of odometry calibration is related to identification of a parameter set that can estimate mobile robot pose in real time with a minimal pose error growth rate. One approach that can be done is an optimization procedure with a criterion that minimizes pose error Ivanjko et al. (2007). In such a procedure firstly mobile robot motion data have to be collected in experiments that distinct the influences of the two mentioned systematic errors. Then an optimization procedure with a criterion that minimizes end pose error can be done resulting with calibration parameters values. Motion data that have to be collected during calibration experiments are mobile robot drive wheel speeds and their sampling times. Crucial for all mentioned methods is measurement of the exact mobile robot start and end pose which is in our case done by a global vision system described in details in Brezak et al. (2008).

3.1.1 Calibration experiments

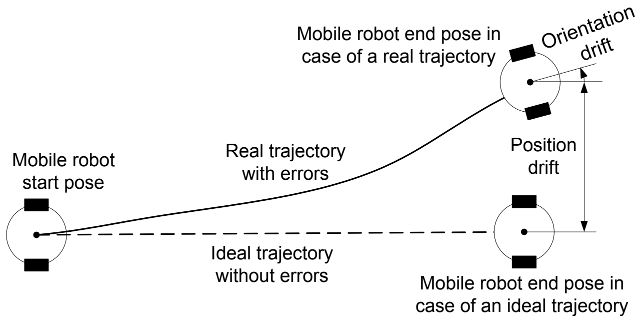

Mobile robot end pose in case of a real trajectory
Orientation drift
Mobile robot start pose
Real trajectory with errors
Position drift
Ideal trajectory without errors
Mobile robot end pose in case of an ideal trajectory

Fig. 4. Straight line experiment

Experiments for optimization of data sets collection must have a trajectory that can gather needed information about both, translational (type B) and rotational (type A) systematic errors. During the experiments drive wheel speeds and sampling time have to be collected, start and end exact mobile robot pose has to be measured. For example, a popular calibration and benchmark trajectory, called UMBmark test Borenstein & Feng (1996), uses a 5 $[m]$ square trajectory performed in both, clockwise and counterclockwise directions. It's good for data collection because it consist of straight parts and turn in place parts but requires a big room.

In Ivanjko et al. (2003) we proposed a set of two trajectories which require significantly less space. First trajectory is a straight line trajectory (Fig. 4), and the second one is a turn in place trajectory (Fig. 5), that has to be done in both directions. Length of the straight line trajectory is $5\,[m]$ like the one square side length in the UMBmark method, and the turn in place experiment is done for $180\,[^\circ]$. This trajectories can be successfully applied to described three parameters expanded kinematic model Ivanjko et al. (2007) with an appropriate optimization criterion.

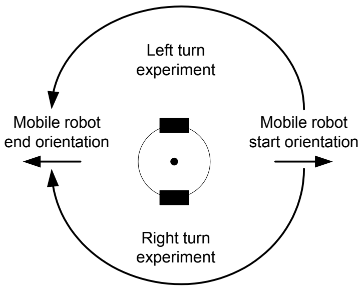


Fig. 5. Turn in place experiments

During experiments collected data were gathered in two groups, each group consisting of five experiments. First (calibration) group of experiments was used for odometry calibration and second (validation) group was used for validation of the obtained calibration parameters. Final calibration parameters values are averages of parameter values obtained from the five collected calibration data sets.

3.1.2 Parameters optimization

Before the optimization process can be started, an optimization criterion I, parameters that will be optimized, and their initial values have to be defined. In our case the optimization criterion is pose error minimum between the mobile robot final pose estimated using the three calibration parameters expanded kinematics model and exact measured mobile robot final pose. Parameters, which values will be changed during the optimization process, are the odometry calibration parameters.

Optimization criterion and appropriate equations that compute the mobile robot final pose is implemented as a m-function in software packet MATLAB. In our case such function consists of three parts: (i) experiment data retrieval, (ii) mobile robot final pose computation using new calibration parameters values, and (iii) optimization criterion value computation. Experiment data retrieval is done by loading needed measurements data from textual files. Such textual files are created during calibration experiments in a proper manner. That means file format has to imitate a ecumenical matrix structure. Numbers that present measurement data that have to be saved in a row are separated using a space sign and a new matrix row is denoted by a new line sign. So data saved in the same row belong to the same time step k. Function inputs are new values of the odometry calibration parameters, and output is new value of the optimization criterion. Function input is computed from the higher lever optimization function using an adequate optimization algorithm. Pseudo code of the here needed optimization m-functions is given in Algorithm 1 where $X(k)$ denotes estimated mobile robot pose.

Algorithm 1 Odometric calibration optimization criterion computation function pseudo code

```
Require: New calibration parameters values {Function input parameters}
Require: Measurement data: drive wheel velocities, time data, exact start and final mobile
    robot pose {Measurement data are loaded from an appropriately created textual file}
Require: Additional calibration parameters values {Parameters $k_1$ and $k_2$ for $k_3$ computation
    and vice versa}
 1: $\omega_L, \omega_R \Leftarrow$ drive wheel velocities data file
 2: $T \Leftarrow$ time data file
 3: $X_{start}, X_{final} \Leftarrow$ exact start and final mobile robot pose
 4: repeat
 5:     $X(k+1) = X(k) + \Delta X(k)$
 6: until experiment measurement data exist
 7: compute new optimization criterion value
 8: return Optimization criterion value
```

In case of the expanded kinematic model with three parameters both experiments (straight line trajectory and turn in place) data and respectively two optimization m-functions are needed. Optimization is so done iteratively. Facts that calibration parameters k_1 and k_2 have the most influence on the straight line experiment and calibration parameter k_3 has the most influence on the turn in place experiment are exploited. Therefore, first optimal values of calibration parameters k_1 and k_2 are computed using collected data from the straight line experiment. Then optimal value of calibration parameter k_3 is computed using so far known values of calibration parameters k_1 and k_2, and collected data from the turn in place experiment. Whence the turn in place experiment is done in both directions, optimization procedure is done for both directions and average value of k_3 is used for the next iteration. We found out that two iterations were enough. Best optimization criterion for the expanded kinematic model with three parameters was minimization of the mobile robot final orientations differences. This can be explained by the fact that the orientation step depends on all three calibration parameters as given with (7) and (18). Mathematically used optimization criterion can be expressed as:

$$I = \Theta_{est} - \Theta_{exact}, \tag{19}$$

where Θ_{est} denotes estimated mobile robot final orientation $[^\circ]$, and Θ_{exact} exact measured mobile robot final orientation $[^\circ]$. Starting calibration parameters values were set to 1.0. Such calibration parameters value denotes usage of mobile robot nominal kinematics model.
Above described optimization procedure is done using the MATLAB Optimization Toolbox *** (2000). Appropriate functions that can be used depend on the version of MATLAB Optimization Toolbox and all give identical results. We successfully used the following functions: `fsolve`, `fmins`, `fminsearch` and `fzero`. These functions use the Gauss-Newton non-linear optimization method or the unconstrained nonlinear minimization Nelder-Mead method. It has to be noticed here that `fmins` and `fminsearch` functions search for a minimum m-function value and therefore absolute minimal value of the orientation difference has to be used. Except mentioned MATLAB Optimization Toolbox functions other optimization algorithms can be used as long they can accept or solve a minimization problem. When mentioned optimization functions are invoked, they call the above described optimization m-function with new calibration parameters values. Before optimization procedure is started

appropriate optimization m-function has to be prepared, which means exact experiments data have to be loaded and correct optimization criterion has to be used.

3.1.3 Experimental setup for odometry calibration

In this section experimental setup for odometry calibration is described. Main components, presented in Fig. 6 are: differential drive mobile robot with an on-board computer, camera connected to an off-board computer, and appropriate room for performing needed calibration experiments i.e. trajectories. Differential drive mobile robot used here was a Pioneer 2DX from *MOBILE*ROBOTS. It was equipped with an on-board computer from VersaLogic including a WLAN communication connection. In order to accurately and robustly measure the exact pose of calibrated mobile robot by the global vision system, a special patch (Fig. 7) is designed, which should be placed on the top of the robot before the calibration experiment.

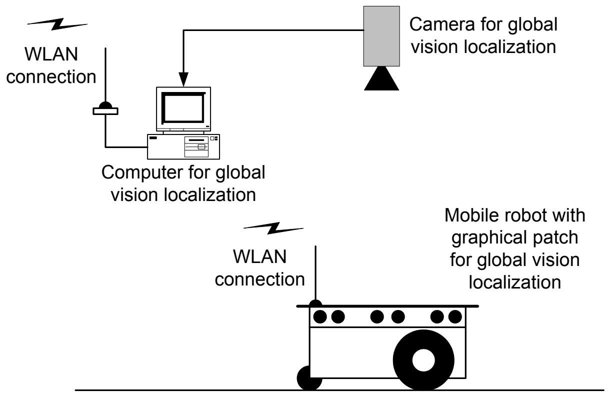

WLAN connection
Camera for global vision localization
Computer for global vision localization
WLAN connection
Mobile robot with graphical patch for global vision localization

Fig. 6. Experimental setup for odometry calibration based on global vision

Software application for control of the calibration experiments, measurement of mobile robot start and end pose, and computation of calibration parameters values is composed from two parts: one is placed (run) on the mobile robot on-board computer and the other one on the off-board computer connected to the camera. Communication between these two application parts is solved using a networking library ArNetworking which is a component of the mobile robot control library ARIA *** (2007). On-board part of application gathers needed drive wheel speeds measurements, sampling time values, and control of the mobile robot experiment trajectories. Gathered data are then send, at the end of each performed experiment, to the off-board part of application. The later part of application decides which particular experiment has to be performed, starts a particular calibration experiment, and measures start and end mobile robot poses using the global vision camera attached to this computer. After all needed calibration experiments for the used calibration method are done, calibration parameters values are computed.

Using described odometry calibration method following calibration parameters values have been obtained: $k_1 = 0.9977$, $k_2 = 1.0023$, and $k_3 = 1.0095$. From the calibration parameters values it can be concluded that used mobile robot has a system error that causes it to slightly turn left when a straight-forward trajectory is performed. Mobile robot odometric system also overestimates its orientation resulting in k_3 value greater then 1.0.

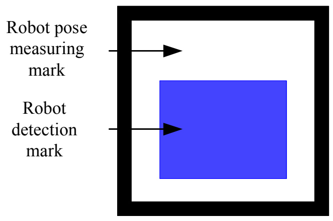


Fig. 7. Mobile robot patch used for pose measurements

3.2 Camera calibration

Camera calibration in the context of threedimensional (3D) machine vision is the process of determining the internal camera geometric and optical characteristics (intrinsic parameters) or the 3D position and orientation of the camera frame relative to a certain world coordinate system (extrinsic parameters) based on a number of points whose object coordinates in the world coordinate system ($X_i, i = 1, 2, \cdots, N$) are known and whose image coordinates ($x_i, i = 1, 2, \cdots, N$) are measured. It is a nonlinear optimization problem (20) whose solution is beyond the scope of this chapter. In our work perspective camera's parameters were determined by off-line camera calibration procedure described in Tsai (1987).

$$min \sum_{i=1}^{N} \left(SC^{-1} X_i - x_i \right)^2 \tag{20}$$

By this method with non-coplanar calibration target and full optimization, obtained were the following intrinsic parameters for SONY EVI-D31 pan-tilt-zoom analog camera and framegrabber with image resolution 320x240:

$$\alpha_x = \alpha_y = 379 \text{ [pixel]},$$
$$x_0 = 165.9 \text{ [pixel]}, \; y_0 = 140 \text{ [pixel]}.$$

4. Sonar based localization

A challenge of mobile robot localization using sensor fusion is to weigh its pose (i.e. mobile robot's state) and sonar range reading (i.e. mobile robot's output) uncertainties to get the optimal estimate of the pose, i.e. to minimize its covariance. The Kalman filter Kalman (1960) assumes the Gaussian probability distributions of the state random variable such that it is completely described with the mean and covariance. The optimal state estimate is computed in two major stages: time-update and measurement-update. In the time-update, state prediction is computed on the base of its preceding value and the control input value using the motion model. Measurement-update uses the results from time-update to compute the output predictions with the measurement model. Then the predicted state mean and covariance are corrected in the sense of minimizing the state covariance with the weighted difference between predicted and measured outputs. In succession, motion and measurement models needed for the mobile robot sensor fusion are discussed, and then EKF and UKF algorithms for mobile robot pose tracking are presented. Block diagram of implemented Kalman filter based localization is given in Fig. 8.

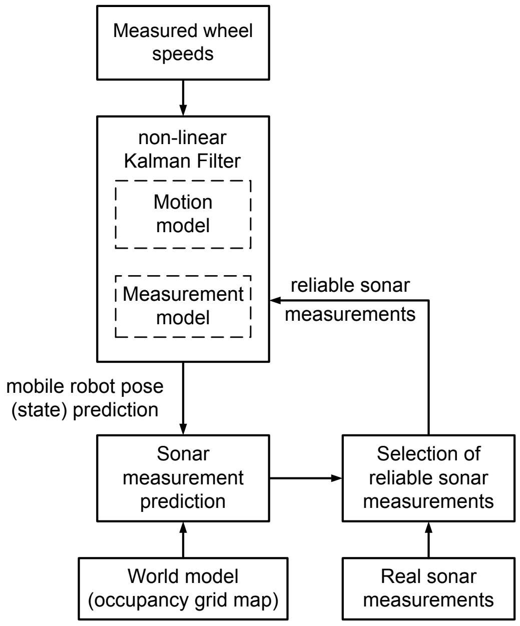

Measured wheel speeds
non-linear Kalman Filter
Motion model
Measurement model
reliable sonar measurements
mobile robot pose (state) prediction
Sonar measurement prediction
Selection of reliable sonar measurements
World model (occupancy grid map)
Real sonar measurements

Fig. 8. Block diagram of non-linear Kalman filter localization approaches.

4.1 Occupancy grid world model

In mobile robotics, an occupancy grid is a two dimensional tessellation of the environment map into a grid of equal or unequal cells. Each cell represents a modelled environment part and holds information about the occupancy status of represented environment part. Occupancy information can be of probabilistic or evidential nature and is often in the numeric range from 0 to 1. Occupancy values closer to 0 mean that this environment part is free, and occupancy values closer to 1 mean that an obstacle occupies this environment part. Values close to 0.5 mean that this particular environment part is not yet modelled and so its occupancy value is unknown. When an exploration algorithm is used, this value is also an indication that the mobile robot has not yet visited such environment parts. Some mapping methods use this value as initial value. Figure 9 presents an example of ideal occupancy grid map of a small environment. Left part of Fig. 9 presents outer walls of the environment and cells belonging to an empty occupancy grid map (occupancy value of all cells set to 0 and filled with white color). Cells that overlap with environment walls should be filled with information that this environment part is occupied (occupancy value set to 1 and filled with black color as it can be seen in the right part of Fig. 9). It can be noticed that cells make a discretization of the environment, so smaller cells are better for a more accurate map. Drawback of smaller cells usage is increased memory consumption and decreased mapping speed because occupancy information in more cells has to be updated during the mapping process. A reasonable tradeoff between memory consumption, mapping speed, and map accuracy can be made with cell size of 10 [cm] x 10 [cm]. Such a cell size is very common when occupancy grid maps are used and is used in our research too.

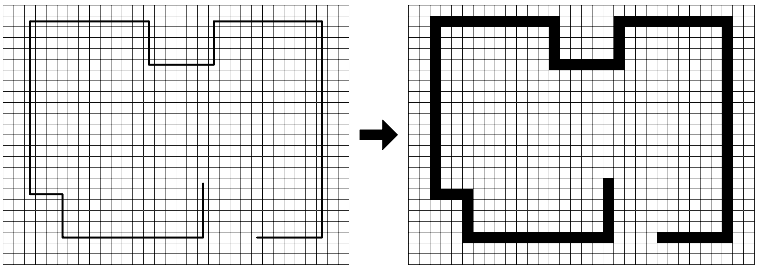
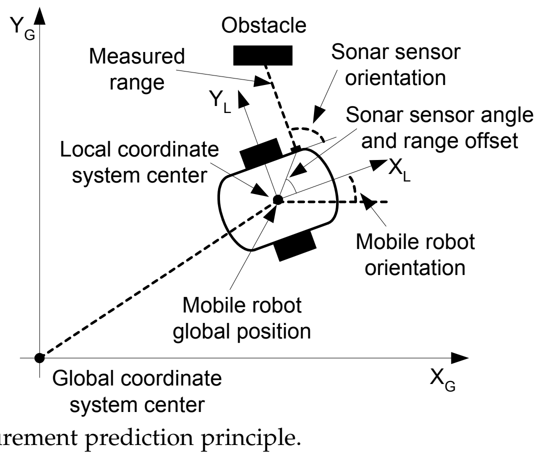


Fig. 9. Example of occupancy grid map environment

Obtained occupancy grid map given in the right part of Fig. 9 does not contain any unknown space. A map generated using real sonar range measurement will contain some unknown space, meaning that the whole environment has not been explored or that during exploration no sonar range measurement defined the occupancy status of some environment part.
In order to use Kalman filter framework given in Fig. 8 for mobile robot pose estimation, prediction of sonar sensor measurements has to be done. The sonar feature that most precise measurement information is concentrated in the main axis of the sonar main lobe is used for this step. So range measurement prediction is done using one propagated beam combined with known local sensor coordinates and estimated mobile robot global pose. Measurement prediction principle is depicted in Fig. 10.


Y_G
Obstacle
Measured range
Sonar sensor orientation
Y_L
Sonar sensor angle and range offset
Local coordinate system center
X_L
Mobile robot orientation
Mobile robot global position
Global coordinate system center
X_G
Fig. 10. Sonar measurement prediction principle.

It has to be noticed that there are two sets of coordinates when measurement prediction is done. Local coordinates defined to local coordinate system (its axis are denoted with X_L and Y_L in Fig. 10) that is positioned in the axle center of the robot drive wheels. It moves with the robot and its x-axis is always directed into the current robot motion direction. Sensors coordinates are defined in this coordinate system and have to be transformed in the global coordinate system center (its axis are denoted with X_G and Y_G in Fig. 10) to compute relative distance between the sonar sensor and obstacles. This transformation for a particular sonar sensor is given by the following equations:

$$S_{XG} = x + S_{offD} \cdot cos\left(S_{off\Theta} + \Theta\right), \tag{21}$$

$$S_{YG} = y + S_{offD} \cdot sin\left(S_{off\Theta} + \Theta\right), \tag{22}$$

$$S_{\Theta G} = \Theta + S_{sens\Theta}, \tag{23}$$

where coordinates x and y present mobile robot global position $[mm]$, Θ mobile robot global orientation $[^\circ]$, coordinates S_{XG} and S_{YG} sonar sensor position in global coordinates $[mm]$, $S_{\Theta G}$ sonar sensor orientation in the global coordinate system frame $[^\circ]$, S_{offD} sonar sensor distance from the center of the local coordinate system $[mm]$, $S_{off\Theta}$ sonar sensor angular offset towards local coordinate system $[^\circ]$, and $S_{\Theta G}$ sonar sensor orientation towards the global coordinate system $[^\circ]$.
After above described coordinate transformation is done, start point and direction of the sonar acoustic beam are known. Center of the sound beam is propagated from the start point until it hits an obstacle. Obtained beam length is then equal to predicted sonar range measurement. Whence only sonar range measurements smaller or equal then 3.0 m are used, measurements with a predicted value greater then 3.0 m are are being discarded. Greater distances have a bigger possibility to originate from outliers and are so not good for pose correction.

4.2 EKF localization

The motion model represents the way in which the current state follows from the previous one. State vector is expressed as the mobile robot pose, $\mathbf{x}_k = [x_k\ y_k\ \Theta_k]^T$, with respect to a global coordinate frame, where k denotes the sampling instant. Its distribution is assumed to be Gaussian, such that the state random variable is completely determined with a 3×3 covariance matrix $\mathbf{P}_k$ and the state expectation (mean, estimate are used as synonyms). Control input, $\mathbf{u}_k$, represents the commands to the robot to move from time step k to $k+1$. In the motion model $\mathbf{u}_k = [D_k\ \Delta\Theta_k]^T$ represents translation for distance D_k followed by a rotation for angle $\Delta\Theta_k$. The state transition function $\mathbf{f}(\cdot)$ uses the state vector at the current time instant and the current control input to compute the state vector at the next time instant:

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_k), \tag{24}$$

where $\mathbf{v}_k = \left[v_{1,k}\ v_{2,k}\right]^T$ represents unpredictable process noise, that is assumed to be Gaussian with zero mean, ($\mathbb{E}\{\mathbf{v}_k\} = [0\ 0]^T$), and covariance $\mathbf{Q}_k$. With $\mathbb{E}\{\cdot\}$ expectation function is denoted. Using (1) to (3) the state transition function becomes:

$$\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_k) = \begin{bmatrix} x_k + (D_k + v_{1,k}) \cdot \cos(\Theta_k + \Delta\Theta_k + v_{2,k}) \\ y_k + (D_k + v_{1,k}) \cdot \sin(\Theta_k + \Delta\Theta_k + v_{2,k}) \\ \Theta_k + \Delta\Theta_k + v_{2,k} \end{bmatrix}. \tag{25}$$

The process noise covariance $\mathbf{Q}_k$ was modelled on the assumption of two independent sources of error, translational and angular, i.e. D_k and $\Delta\Theta_k$ are added with corresponding uncertainties. The expression for $\mathbf{Q}_k$ is:

$$\mathbf{Q}_k = \begin{bmatrix} \sigma_D^2 & 0 \\ 0 & \Delta\Theta_k^2 \sigma_{\Delta\Theta}^2 \end{bmatrix}, \tag{26}$$

where σ_D^2 and $\sigma_{\Delta\Theta}^2$ are variances of D_k and $\Delta\Theta_k$, respectively.
The measurement model computes the range between an obstacle and the axle center of the robot according to a measurement function Lee (1996):

$$h_i(\mathbf{x}, p_i) = \sqrt{(x_i - x)^2 + (y_i - y)^2}, \tag{27}$$

where $p_i = (x_i, y_i)$ denotes the point (occupied cell) in the world model detected by the ith sonar. The sonar model uses (27) to relate a range reading to the obstacle that caused it:

$$z_{i,k} = h_i(\mathbf{x}_k, p_i) + w_{i,k}, \tag{28}$$

where $w_{i,k}$ represents the measurement noise (Gaussian with zero mean and variance $r_{i,k}$) for the ith range reading. All range readings are used in parallel, such that range measurements $z_{i,k}$ are simply stacked into a single measurement vector $\mathbf{z}_k$. Measurement covariance matrix $\mathbf{R}_k$ is a diagonal matrix with the elements $r_{i,k}$. It is to be noted that the measurement noise is additive, which will be beneficial for UKF implementation.
EKF is the first sensor fusion based mobile robot pose tracking technique presented in this paper. Detailed explanation of used EKF localization can be found in Ivanjko et al. (2004) and in the sequel only basic equations are presented. Values of the control input vector $\mathbf{u}_{k-1}$ computed from wheels' encoder data are passed to the algorithm at time k such that first time-update is performed obtaining the prediction estimates, and then if new sonar readings are available those predictions are corrected. Predicted (prior) state mean $\hat{\mathbf{x}}_k^-$ is computed in single-shot by propagating the state estimated at instant $k-1$, $\hat{\mathbf{x}}_{k-1}$ through the true nonlinear odometry mapping:

$$\hat{\mathbf{x}}_k^- = \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbb{E}\{\mathbf{v}_{k-1}\}). \tag{29}$$

The covariance of the predicted state $\mathbf{P}_k^-$ is approximated with the covariance of the state propagated through a linearized system from (24):

$$\mathbf{P}_k^- = \nabla\mathbf{f}_{\mathbf{x}} \mathbf{P}_{k-1} \nabla\mathbf{f}_{\mathbf{x}}^T + \nabla\mathbf{f}_{\mathbf{u}} \mathbf{Q}_k \nabla\mathbf{f}_{\mathbf{u}}^T, \tag{30}$$

where $\nabla\mathbf{f}_{\mathbf{x}} = \nabla\mathbf{f}_{\mathbf{x}}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbb{E}\{\mathbf{v}_{k-1}\})$ is the Jacobian matrix of $\mathbf{f}$ with respect to $\mathbf{x}$, while $\nabla\mathbf{f}_{\mathbf{u}} = \nabla\mathbf{f}_{\mathbf{u}}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbb{E}\{\mathbf{v}_{k-1}\})$ is the Jacobian matrix of $\mathbf{f}(\cdot)$ with respect to control input $\mathbf{u}$. It is to be noticed that using (29) and (30) the mean and covariance are accurate only to the first-order of the corresponding Taylor series expansion Haykin (2001). If there are no new sonar readings at instant k or if they are all rejected, measurement update does not occur and the estimate mean and covariance are assigned with the predicted ones:

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^-, \tag{31}$$

$$\mathbf{P}_k = \mathbf{P}_k^-. \tag{32}$$

Otherwise, measurement-update takes place where first predictions of the accepted sonar readings are collected in $\hat{\mathbf{z}}_k^-$ with ith component of it being:

$$\hat{z}_{i,k}^- = h_i(\hat{\mathbf{x}}_k^-, p_i) + E\{w_{i,k}\}. \tag{33}$$

The state estimate and its covariance in time step k are computed as follows:

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \hat{\mathbf{z}}_k^-), \tag{34}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \nabla \mathbf{h}_\mathbf{x})\mathbf{P}_k^-, \tag{35}$$

where $\mathbf{z}_k$ are real sonar readings, $\nabla \mathbf{h}_\mathbf{x} = \nabla \mathbf{h}_\mathbf{x}(\hat{\mathbf{x}}_k^-, \mathbb{E}\{\mathbf{w}_k\})$ is the Jacobian matrix of the measurement function with respect to the predicted state, and $\mathbf{K}_k$ is the optimal Kalman gain computed as follows:

$$\mathbf{K}_k = \mathbf{P}_k^- \nabla \mathbf{h}_\mathbf{x}^T (\nabla \mathbf{h}_\mathbf{x} \mathbf{P}_k^- \nabla \mathbf{h}_\mathbf{x}^T + \mathbf{R}_k)^{-1}. \tag{36}$$

4.3 UKF localization

The second sensor fusion based mobile robot pose tracking technique presented in this chapter uses UKF. UKF was first proposed by Julier et al. Julier & Uhlmann (1996), and further developed by Wan and van der Merwe Haykin (2001). It utilizes the unscented transformation Julier & Uhlmann (1996) that approximates the true mean and covariance of a Gaussian random variable propagated through nonlinear mapping accurate to the inclusively third order of Taylor series expansion for any mapping. Following this, UKF approximates state and output mean and covariance more accurately than EKF and thus superior operation of UKF compared to EKF is expected. UKF was already used for mobile robot localization in Ashokaraj et al. (2004) to fuse several sources of observations, and the estimates were, if necessary, corrected using interval analysis on sonar measurements. Here we use sonar measurements within UKF, without any other sensors except the encoders to capture angular velocities of the drive wheels (motion model inputs), and without any additional estimate corrections.

Means and covariances are in UKF case computed by propagating carefully chosen so called pre-sigma points through the true nonlinear mapping. Nonlinear state-update with non-additive Gaussian process noises in translation D and rotation $\Delta\Theta$ is given in (25). The measurement noise is additive and assumed to be Gaussian, see (28).

The UKF algorithm is initialized ($k = 0$) with $\hat{\mathbf{x}}_0$ and $\mathbf{P}_0$, same as the EKF. In case of non-additive process noise and additive measurement noise, state estimate vector is augmented with means of process noise $\mathbb{E}\{\mathbf{v}_{k-1}\}$ only, thus forming extended state vector $\hat{\mathbf{x}}_{k-1}^a$:

$$\hat{\mathbf{x}}_{k-1}^a = \mathbb{E}[\mathbf{x}_{k-1}^a] = \begin{bmatrix} \hat{\mathbf{x}}_{k-1}^T & \mathbb{E}\{\mathbf{v}_{k-1}\}^T \end{bmatrix}^T. \tag{37}$$

Measurement noise does not have to enter the $\hat{\mathbf{x}}_{k-1}^a$ because of additive properties Haykin (2001). This is very important from implementation point of view since the dimension of output is not known in advance because number of accepted sonar readings varies. Covariance matrix is augmented accordingly forming matrix $\mathbf{P}_{k-1}^a$:

$$\mathbf{P}_{k-1}^a = \begin{bmatrix} \mathbf{P}_{k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_{k-1} \end{bmatrix}. \tag{38}$$

Time-update algorithm in time instant k first requires square root of the $\mathbf{P}_{k-1}^a$ (or lower triangular Cholesky factorization), $\sqrt{\mathbf{P}_{k-1}^a}$. Obtained lower triangular matrix is scaled by the factor γ:

$$\gamma = \sqrt{L + \lambda}, \tag{39}$$

where L represents the dimension of augmented state $\mathbf{x}^a_{k-1}$ ($L = 5$ in this application), and λ is a scaling parameter computed as follows:

$$\lambda = \alpha^2(L + \kappa) - L. \tag{40}$$

Parameter α can be chosen within range $[10^{-4}, 1]$, and κ is usually set to 1. There are $2L + 1$ pre-sigma points, the first is $\hat{\mathbf{x}}^a_{k-1}$ itself, and other $2L$ are obtained by adding to or subtracting from $\hat{\mathbf{x}}^a_{k-1}$ each of L columns of $\gamma\sqrt{\mathbf{P}^a_{k-1}}$, symbolically written as:

$$\mathcal{X}^a_{k-1} = \left[\begin{array}{ccc} \hat{\mathbf{x}}^a_{k-1} & \hat{\mathbf{x}}^a_{k-1} + \gamma\sqrt{\mathbf{P}^a_{k-1}} & \hat{\mathbf{x}}^a_{k-1} - \gamma\sqrt{\mathbf{P}^a_{k-1}} \end{array}\right], \tag{41}$$

where $\mathcal{X}^a_{k-1} = [(\mathcal{X}^x_{k-1})^T \ (\mathcal{X}^v_{k-1})^T]^T$ represents the matrix whose columns are pre-sigma points. All pre-sigma points are processed by the state-update function obtaining matrix $\mathcal{X}^x_{k|k-1}$ of predicted states for each pre-sigma point, symbolically written as:

$$\mathcal{X}^x_{k|k-1} = \mathbf{f}[\mathcal{X}^x_{k-1}, \mathbf{u}_{k-1}, \mathcal{X}^v_{k-1}]. \tag{42}$$

Prior mean is calculated as weighted sum of acquired points:

$$\hat{\mathbf{x}}^-_k = \sum_{i=0}^{2L} W^{(m)}_i \mathcal{X}^x_{i,k|k-1}, \tag{43}$$

where $\mathcal{X}^x_{i,k|k-1}$ denotes the ith column of $\mathcal{X}^x_{k|k-1}$. Weights for mean calculation $W^{(m)}_i$ are given by

$$W^{(m)}_0 = \frac{\lambda}{L + \lambda}, \tag{44}$$

$$W^{(m)}_i = \frac{1}{2(L + \lambda)}, \quad i = 1, \ldots, 2L. \tag{45}$$

Prior covariance matrix $\mathbf{P}^-_k$ is given by

$$\mathbf{P}^-_k = \sum_{i=0}^{2L} W^{(c)}_i [\mathcal{X}^x_{i,k|k-1} - \hat{\mathbf{x}}^-_k][\mathcal{X}^x_{i,k|k-1} - \hat{\mathbf{x}}^-_k]^T, \tag{46}$$

where $W^{(c)}_i$ represent the weights for covariance calculation which are given by

$$W^{(c)}_0 = \frac{\lambda}{L + \lambda} + (1 - \alpha^2 + \beta), \tag{47}$$

$$W^{(c)}_i = \frac{1}{2(L + \lambda)}, \quad i = 1, \ldots, 2L. \tag{48}$$

For Gaussian distributions $\beta = 2$ is optimal.
If there are new sonar readings available at time instant k, predicted readings of accepted sonars for each sigma-point are grouped in matrix $\mathcal{Z}_{k|k-1}$ obtained by

$$\mathcal{Z}_{k|k-1} = \mathbf{h}[\mathcal{X}^x_{k|k-1}, \mathbf{p}] + \mathbb{E}\{\mathbf{w}_k\}, \tag{49}$$

where $\mathbf{p}$ denotes the series of points in the world map predicted to be hit by sonar beams. Predicted readings $\hat{\mathbf{z}}_k^-$ are then

$$\hat{\mathbf{z}}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{Z}_{i,k|k-1}. \tag{50}$$

To prevent the sonar readings that hit near the corner of obstacles to influence on the measurement correction, since their probability distribution cannot be approximated with Gaussian, another threshold comparison were made. These problematic sonar readings are recognized with mean $\hat{z}_{i,k}^-$ that differs from $z_{i,k}$ more than the acceptation threshold amounts, and those are being discarded. Readings covariance is

$$\mathbf{P}_{\mathbf{z}_k} = \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{Z}_{i,k|k-1} - \hat{\mathbf{z}}_k^-][\mathcal{Z}_{i,k|k-1} - \hat{\mathbf{z}}_k^-]^T + \mathbf{R}_k, \tag{51}$$

and state-output cross-covariance matrix is

$$\mathbf{P}_{\mathbf{x}_k \mathbf{z}_k} = \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{X}_{i,k|k-1}^x - \hat{\mathbf{x}}_k^-][\mathcal{Z}_{i,k|k-1} - \hat{\mathbf{z}}_k^-]^T. \tag{52}$$

Kalman gain $\mathbf{K}_k$ is

$$\mathbf{K}_k = \mathbf{P}_{\mathbf{x}_k \mathbf{z}_k} \mathbf{P}_{\mathbf{z}_k}^{-1}. \tag{53}$$

Posterior state covariance is finally calculated as

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}_k \mathbf{P}_{\mathbf{z}_k} \mathbf{K}_k^T. \tag{54}$$

The measurement correction is done as in (34).

5. Monocular vision based localization

In this section, we consider the problem of mobile robot pose estimation using only visual information from a single camera and odometry readings. Focus is on building complex environmental models, fast online rendering and real-time complex and noisy image segmentation. The 3D model of the mobile robot's environment is built using a professional freeware computer graphics tool named Blender and pre-stored in the memory of the robot's on-board computer. Estimation of the mobile robot pose as a stochastic variable is done by correspondences of image lines, extracted using Random Window Randomized Hough Transform line detection algorithm, and model lines, predicted using odometry readings and 3D environment model. The camera model and ray tracing algorithm are also described. Developed algorithms are also experimentally tested using a Pioneer 2DX mobile robot.

5.1 Scene modelling and rendering

Referential model of the environment was built using *Blender*, where vertices, edges (lines) and faces (planes) were used for model notation. An edge is defined with two vertices and a face with three or four vertices. The drawn model is one object in which all vertices, edges and faces are listed. For illustration, in Fig. 11, 3D model of the hallway in which our mobile robot moves is shown. Although the environment model is quite complex, we achieved online execution of the localization algorithms by applying fast rendering. Namely, in each

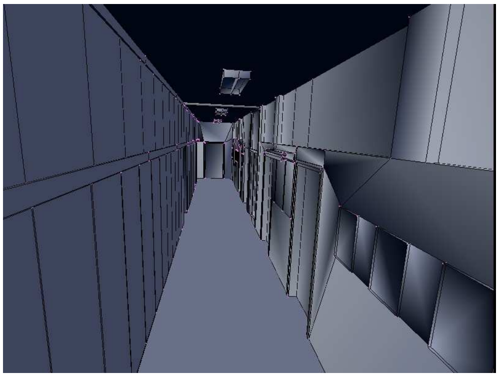
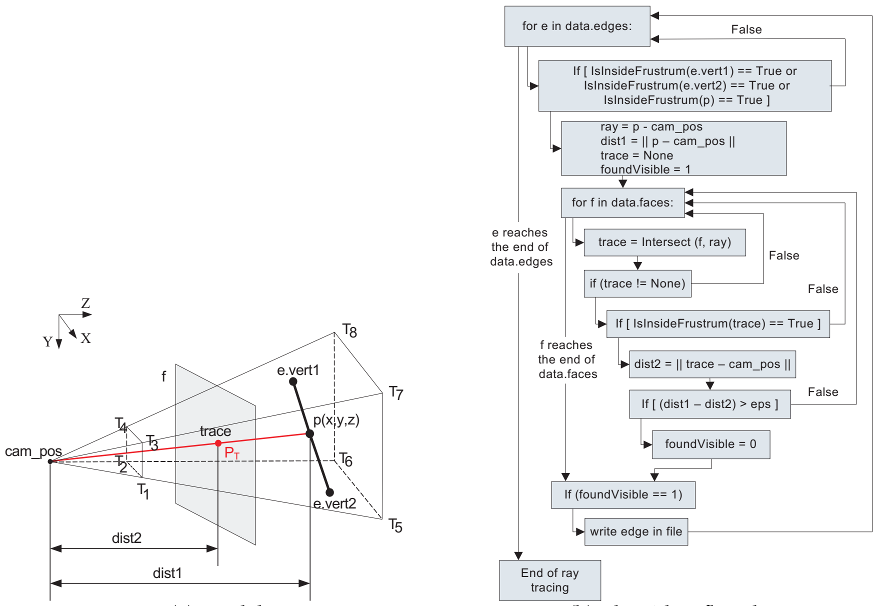


Fig. 11. Hallway 3D model

step we render only the small region enclosed with the camera frustrum and then apply a ray tracing algorithm to solve the occlusion problems.
The above described notation of the scene model, enables us to implement a ray tracing algorithm. The algorithm is organized in two "for" loops, as shown in Fig. 12(b), where the algorithm flowchart is depicted. The first (outer) loop goes over the edge list and the second (inner) loop goes over the face list. The outer loop starts with the function IsInsideFrustrum (point3D), which examines whether observed points are located inside the camera frustrum and discards those that are not in it. Then, for a point p in the frustrum, where p is the point in the middle of the edge determined with two vertices, e.vert1 and e.vert2, as shown in Fig. 12(a), the direction of the vector ray is defined with point p and camera pose (cam_pos).
The inner loop starts with choosing a plane f from the list of faces, and then the function Intersect (f, vector) returns intersection point P_T between the given plane f and direction vector as an output value, or *None* if the intersection doesn't exist. Visible edges are checked by comparing distances from the camera pose to the the point p (dist1) and to intersection point P_T (dist2), see Fig. 12(a). If these two distances do not match, the checked model edge (line) is invisible, and therefore not used in later matching procedure.
Notice the incompleteness of rendering because only edges whose middle point is visible will be rendered visible. That does not affect the accuracy of the later matching algorithm for partially visible model lines because it is done in Hough space where a line is represented with a single point regarding its length. The rendering could produce only smaller number of partially visible lines, but in this case it is not important because there are still enough lines for estimating mobile robot's pose while gaining faster algorithm execution.

5.2 Image segmentation

Mobile robot self-localization requires matching of edge segments in the current camera image and edge segments in the environment model seen from the expected mobile robot pose. In previous section we described line extraction from the environment model, and below we describe the line extraction in the camera image (image segmentation). Image segmentation is done by the Canny edge detector and RWRHT line detector algorithm described in Kälviäinen et al. (1994). The RWRHT is based on Randomized Hough Transformation (RHT), which selects n pixels from the edge image by random sampling to solve n parameters of


Z
Y
X
T8
T7
T6
T5
T4
T3
T2
T1
f
e.vert1
e.vert2
p(x,y,z)
trace
P_T
cam_pos
dist2
dist1
for e in data.edges:
False
If [IsInsideFrustrum(e.vert1) == True or IsInsideFrustrum(e.vert2) == True or IsInsideFrustrum(p) == True]
ray = p - cam_pos
dist1 = || p – cam_pos ||
trace = None
foundVisible = 1
for f in data.faces:
trace = Intersect (f, ray)
if (trace != None)
If [IsInsideFrustrum(trace) == True]
dist2 = || trace – cam_pos ||
If [(dist1 – dist2) > eps]
foundVisible = 0
If (foundVisible == 1)
write edge in file
e reaches the end of data.edges
f reaches the end of data.faces
End of ray tracing

(a) model (b) algorithm flowchart

Fig. 12. Ray tracing

a curve, and then accumulates only one cell in parameter space. Detected curves are those whose accumulator cell is greater then predefined threshold. The RWRHT is an extension of the RHT on complex and noisy images that applies RHT to a limited neighborhood of edge pixels. The benefit is the reduction of the computational power and memory resources. The pseudo-code of the RWHT is written in Algorithm 2.

5.3 Mobile robot pose estimation

Once the correspondences have been established between image lines and model lines seen from the current mobile robot pose, we could update the mobile robot pose by applying an appropriate estimation technique. In most cases, linearized system and measurement equations and Gaussian noise in states and measurements are satisfactory approximations. Therefore, we apply Extended Kalman Filter (EKF) Welch & Bishop (2000), which is an optimal estimator under the above assumptions.

The state vector that is to be estimated is the mobile robot pose **p**. Introducing uncertainty in the equations (1), (2) and (3) as the zero mean Gaussian additive noise, the state equations are obtained:

$$\mathbf{p_{n+1}} = \mathbf{f}\left[\mathbf{p_n}, v(n), \omega(n)\right] + w(n), \tag{55}$$

where $w \sim \mathcal{N}(0, Q)$.

Algorithm 2 The Random Window RHT Algorithm

```
1: $D \Leftarrow$ all edge points in binary edge picture
2: $d_i \Leftarrow$ randomly selected point from set $D$
3: $m \Leftarrow$ randomly selected window size where $m_{min} \leq m \leq m_{max}$
4: $W \Leftarrow$ pixel data set of $m \times m$ neighborhood of the point $d_i$
5: repeat
6:     RHT algorithm on the set $W$
7: until maximum R times
8: if accumulator cell $\geq$ threshold then
9:     corresponding parameters are the parameters of detected curve
10: else
11:     goto 2
12: end if
```

Measurement is a set $\mathcal{S}$ of pairs "model line - image line":

$$\begin{aligned} \mathcal{S} = \{\{m,i\} | m &= \text{visible model line}, \\ i &= \text{image line - perspective projection of m}\}. \end{aligned} \tag{56}$$

The straight line in the world coordinate system which passes through the point (x_0, y_0, z_0) and has direction coefficients (a, b, c) is given by:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} u + \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}, \; u \in \Re. \tag{57}$$

The straight line in image plane is given by

$$x \; cos\gamma \; + y \; sin\gamma \; = \rho, \quad \rho \geq 0, \quad \gamma \in [0, 2\pi], \tag{58}$$

where ρ and γ are the Hough space parameters of the line.
Let by applying perspective projection transformation P to a 3D model line we obtain 2D straight line m and let its pair line i in the image be defined with the Hough space parameters (ρ, γ). The projection of the 3D point (x_0, y_0, z_0) lies on the image line i and direction coefficients of m and i lines are the same if the following conditions are fulfilled:

$$
\begin{aligned}
z_1 &= W \begin{bmatrix} \rho\, cos\gamma \\ \rho\, sin\gamma \\ 1 \end{bmatrix} - P \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} = 0, & (59) \\
z_2 &= \begin{bmatrix} -sin\gamma \\ cos\gamma \\ 0 \end{bmatrix} - \frac{P \begin{bmatrix} a \\ b \\ c \\ 0 \end{bmatrix}}{\left\| P \begin{bmatrix} a \\ b \\ c \\ 0 \end{bmatrix} \right\|} = 0, & (60) \\
\mathbf{z} &= [z_1, z_2]^\top . & (61)
\end{aligned}
$$

Equations (59) and (60) are measurement equations and are used to correct mobile robot pose with each correct match. Uncertainty in mobile robot pose means uncertainty in model lines and is propagated to the Hough space in the same way as in Kosaka & Kak (1992), so we will not go into details but only explain the main concept. For differentials of the equation (15) we obtain

$$\delta\mathbf{X} = \begin{bmatrix} \delta X \\ \delta Y \end{bmatrix} = J_{\mathcal{P}}(\mathbf{M}, \hat{\mathbf{p}})\, \delta\mathbf{p}\,, \tag{62}$$

where $J_{\mathcal{P}}(\hat{\mathbf{p}})$ is Jacobian of perspective projection of the end point of 3D model line M taken at expected values of random vector $\mathbf{p}$. Notice that from this point on, we are using first order Taylor approximations of nonlinear transformations. Covariance matrix related to pixel coordinates of a single model line point is given by:

$$\Sigma_X = E[\delta\mathbf{X}\delta\mathbf{X}^\top] = J_{\mathcal{P}}(\mathbf{M}, \hat{\mathbf{p}})\; \Sigma_p\; J_{\mathcal{P}}(\mathbf{M}, \hat{\mathbf{p}})^\top, \tag{63}$$

where Σ_p is covariance matrix of mobile robot pose $\mathbf{p}$. So, at this moment, we have determined uncertainty convex hull in which the probability of finding corresponding image line is the highest. Furthermore, we can apply Hough transform $\mathcal{H}$ to that line segment, which would lead to point representation $\hat{h} = (\rho, \gamma)$ of the line on which the segment coincides with elliptical uncertainty region defined by the Mahalanobious distance and covariance matrix. If $J_{\mathcal{H}}$ denotes Jacobian of the Hough transform $\mathcal{H}$ with respect to variables ρ and γ we can write:

$$\begin{bmatrix} \delta\rho \\ \delta\gamma \end{bmatrix} = J_{\mathcal{H}} \begin{bmatrix} \delta X_1 \\ \delta Y_1 \\ \delta X_2 \\ \delta Y_2 \end{bmatrix}, \tag{64}$$

$$\Sigma_{\rho\gamma} = J_{\mathcal{H}} \begin{bmatrix} J_{\mathcal{P}}(\mathbf{M_1}, \hat{\mathbf{p}}) & 0 \\ 0 & J_{\mathcal{P}}(\mathbf{M_2}, \hat{\mathbf{p}}) \end{bmatrix} \Sigma_p \begin{bmatrix} J_{\mathcal{P}}(\mathbf{M_1}, \hat{\mathbf{p}}) & 0 \\ 0 & J_{\mathcal{P}}(\mathbf{M_2}, \hat{\mathbf{p}}) \end{bmatrix}^\top J_{\mathcal{H}}^\top. \tag{65}$$

We limit the search for image lines to uncertainty region in the Hough space $\subset \Re^2$ determined by the constraint:

$$(\mathbf{h} - \hat{\mathbf{h}})\Sigma_{\rho\gamma}^{-1}(\mathbf{h} - \hat{\mathbf{h}})^T \leq 2.$$

This rises a matching problem if one model line has more then one image candidate, but the problem was solved in Kosaka & Kak (1992) and Aider et al. (2005).
There are also quantization error, noise in camera image and error in edge detection and image segmentation which have been approximated by Gaussian variable $\xi \sim \mathcal{N}(0, V)$ and included in the EKF equations as the measurement noise. Finally, the equations of the implemented EKF are:
a priori update:

$$\begin{aligned} \hat{\mathbf{p}}_{\mathbf{n+1}} &= f\left[\hat{\mathbf{p}}_n, v(n), \omega(n)\right], & (66)\\ \hat{\mathbf{z}}_{\mathbf{n}} &= \mathbf{z}\left[\hat{\mathbf{p}}_{\mathbf{n}}, \hat{\mathbf{h}}(i)_{i\to m}\right], & (67)\\ \Sigma_{p_{n+1|n}} &= A\,\Sigma_{p_n}\,A^\top + Q, \quad A = \partial\mathbf{f}/\partial\mathbf{p}|_{\mathbf{p}=\hat{\mathbf{p}}}, & (68) \end{aligned}$$

a posteriori update:

$$\begin{aligned} K_{n+1} &= \Sigma_{p_{n+1|n}}\,H^\top\left[H\,\Sigma_{p_{n+1|n}}\,H^\top + R\right], & (69)\\ \mathbf{p}_{\mathbf{n}+1} &= \hat{\mathbf{p}}_{\mathbf{n+1}} + K_{n+1}(\mathbf{z} - \hat{\mathbf{z}}_{\mathbf{n}}), & (70)\\ \Sigma_{p_{n+1}} &= (I - K_{n+1}\,H)\Sigma_{p_{n+1|n}}, & (71) \end{aligned}$$

where $H = \frac{\partial\mathbf{z}}{\partial\mathbf{p}}|_{\mathbf{p}=\hat{\mathbf{p}}}$ and $R = \frac{\partial\mathbf{z}}{\partial\mathbf{h}}|_{\mathbf{h}=\hat{\mathbf{h}}} \cdot V \cdot \frac{\partial\mathbf{z}}{\partial\mathbf{h}}^\top|_{\mathbf{h}=\hat{\mathbf{h}}}$.

6. Experimental results

This section presents obtained experimental results including description of experimental setup and experiment conditions. First are presented results obtained using sonar sensors and after that results obtained using monocular-vision. Section ends with comments on obtained results.

6.1 Experimental setup description

Experiments are performed using a Pioneer 2DX mobile robot from MobileRobots. Its configuration is presented in Fig. 13 only that in localization experiments monocular camera was used insted of depicted stereo one. Used sensors are encoders for odometry, sonars, mono-camera and a laser range finder. Laser range finder was used only for a comparison purpose as a sensor that enables a more accurate localization than the sonars or mono-camera. It is combined with a Monte-Carlo algorithm Konolige (1999) implemented as standard localization solution in the mobile robot control software. Used camera for monocular-vision localization is a SONY EVI-D31 pan-tilt-zoom analog camera. Laser sensor is a SICK LMS-200, and sonars are Polaroid 6500 sensors.
Experimental environment including trajectory traversed by the mobile robot is presented in Fig. 14. Global coordinate system is depicted on the left side. It's a hallway with several door niches. Mobile robot movement started in one corridor end and ended when it reached other corridor end. Trajectory length is approximately 20 [m] and is generated using a gradient based algorithm described in Konolige (2000). Obstacle avoidance was also active during all experiments.

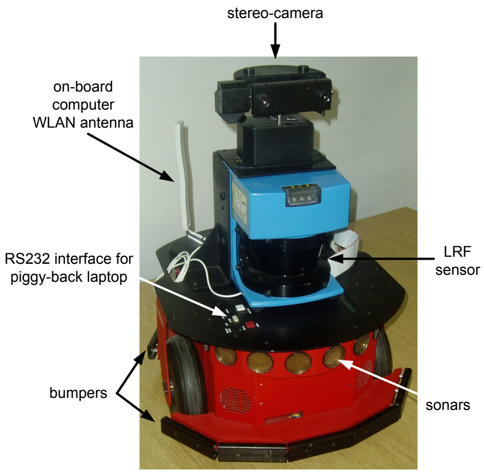

stereo-camera
on-board computer WLAN antenna
RS232 interface for piggy-back laptop
LRF sensor
bumpers
sonars

Fig. 13. Pioneer 2DX mobile robot

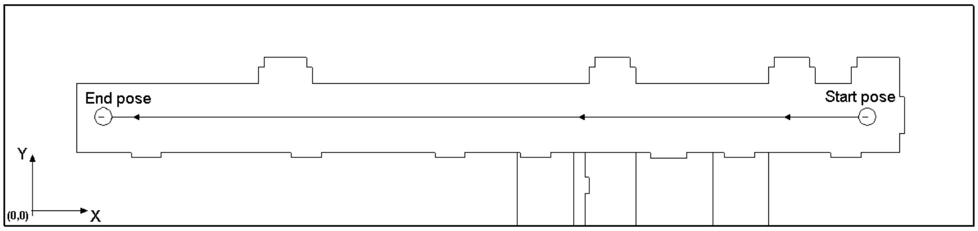


Fig. 14. Mobile robot trajectory for experimental evaluation.

Initial robot's pose and its covariance was in all experiments set to (orientation given in radians):

$$\mathbf{p_0} = \begin{bmatrix} 210 \text{ dm} & 9.25 \text{ dm} & \pi \end{bmatrix}^\top,$$

$$\Sigma_{\mathbf{p_0}} = \begin{bmatrix} 0.3000 & 0 & 0 \\ 0 & 0.3000 & 0 \\ 0 & 0 & 0.0080 \end{bmatrix},$$

which means about 0.55 $[dm]$ standard deviation in p_x and p_y and about 5 $[^\circ]$ standard deviation in robot orientation. In every experiment mobile robot start pose was manually set according to marks on the hallway floor, so given initial pose covariance was set to cover start pose setting error margins. Experiments show that implemented self-localization algorithms were able to cope with that starting pose error margins.
During all experiments relevant measurements data were collected and saved for obtained localization quality evaluation. Saved data included mobile robot drive wheel speeds, sampling period, sonar range measurements, camera images, evaluated self-localization algorithm estimated pose and Monte Carlo based localization results. Pose obtained using Monte Carlo

algorithm and laser range finder sensor was then used as the more accurate i.e. exact mobile robot pose for comparison.
If localization was only done by odometry, results would be like shown on Fig. 15, i.e. pose error would monotonically increase over time. It is obvious that mobile robot cannot perform its tasks without any pose correction.

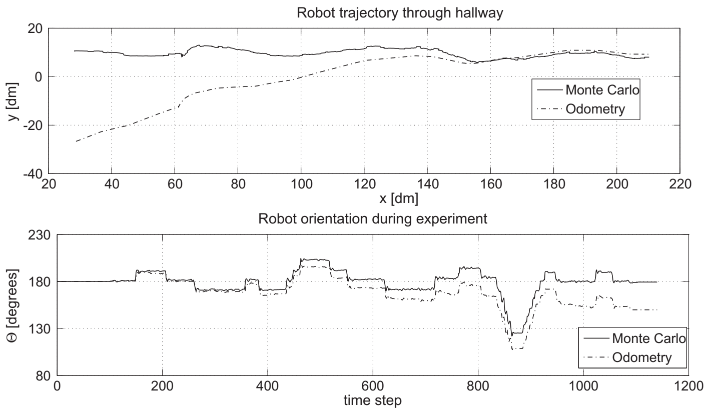


Fig. 15. Robot trajectory method comparison: solid - Monte Carlo localization, dash-dot - odometry

6.2 Sonar localization results

Figures 16 and 17 present results obtained using sonar sensors. Solid line denotes Monte Carlo localization with laser range finder and doted line denotes sonar sensor based localization results. Both figures consist of two parts. Upper part presents mobile robot trajectory i.e. its position and lower part presents mobile robot orientation change.
As mentioned before two sonar sensor based localization approaches were implemented. EKF based results are presented in Fig. 16 and UKF based results are presented in Fig. 17. Both implementations use calibrated odometry as the motion model to increase localization accuracy. An occupancy grid model is used for sonar sensor measurement prediction. Whence occupancy grid size is 100 $[mm]$ x 100 $[mm]$, localization accuracy is expected to be in this range.
Figures given in this section show that sonar sensors can be effectively used for self-localization with accuracy in range of used occupancy model grid size. It can be noticed that EKF ends with a pose estimation with bigger pose corrections. This feature arises from the first order linearization done be the EKF. In case of the UKF corrections are of smaller value as expected. More accurate pose estimation of the UKF can be proved by computing pose error values on the whole traversed path. Pose error is computed as difference between estimated trajectory and referent Monte Carlo pose. With the EKF maximal position error is 3.2 $[dm]$, and maximal orientation error is 6.8 $[^\circ]$, while with the UKF their values are 2.5 $[dm]$, and 3.7 $[^\circ]$. These values are important when self-localization algorithms are used for longer trajectories. A bigger maximal pose error indicates a greater probability that mobile robot will loose its pose indicating a necessary global pose correction.

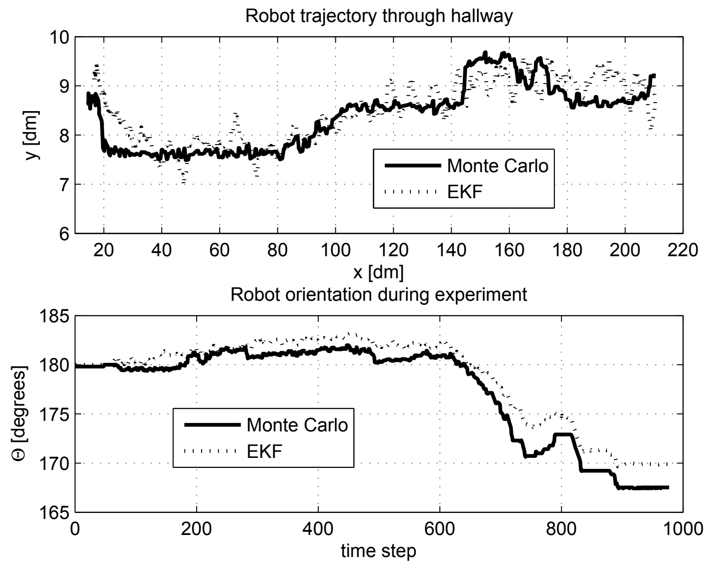


Fig. 16. Robot trajectory method comparison: solid - Monte Carlo localization, dots - EKF method.

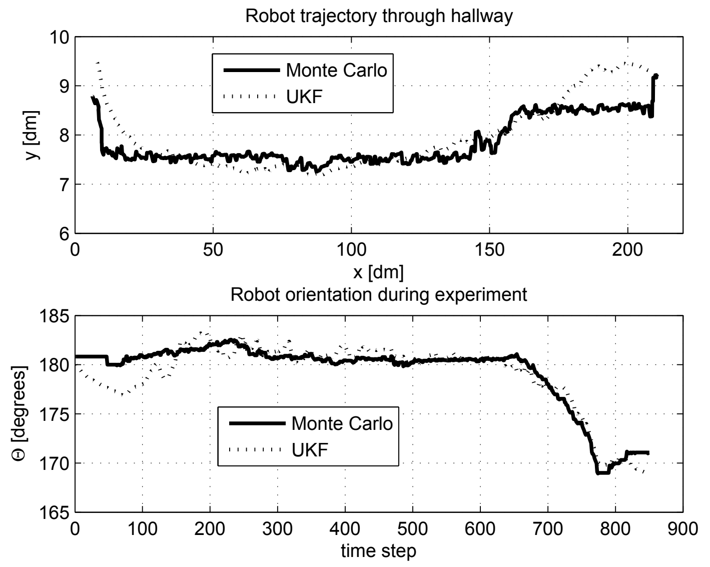


Fig. 17. Robot trajectory method comparison: solid - Monte Carlo localization, dots - UKF method.

6.3 Monocular-vision localization results

Similar as in the sonar localization experiment the mobile robot was given navigational commands to drive along a hallway and to collect odometry data and images from camera fixed to it at multiples of the discretization time. Figure 18 shows line segments superimposed to the camera view. Very good robustness to change of illumination and noise in camera image can be noticed. Figure 19 shows rendered hallway model superimposed to the camera view, before any correction was done. After image acquisition and model rendering, the off-line optimal matching of rendered image lines and lines extracted from the camera image was done. Obtained pairs and rendered model from corrected camera pose are shown in Fig. 20. Updated mobile robot start pose and its covariance was (orientation given in radians):

$$\mathbf{p_0} = \begin{bmatrix} 209.946 \text{ dm} & 9.2888 \text{ dm} & 3.1279 \end{bmatrix}^\top$$

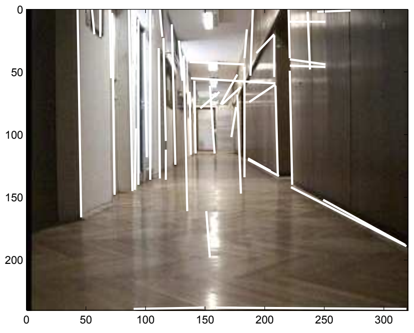


Fig. 18. Superposition of camera view and line segments extracted by RWRHT method

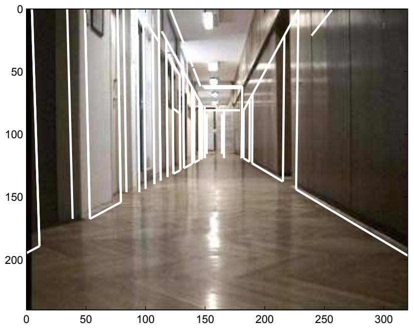


Fig. 19. Superposition of camera view and rendered model before correction

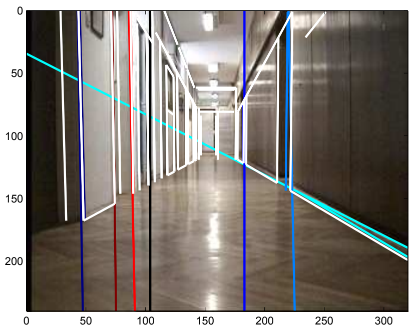


Fig. 20. Superposition of camera view and rendered model after correction. Line pairs that were matched and used as measurement are drawn with different colors for each pair.

$$\Sigma_{\mathbf{p_0}} = \begin{bmatrix} 0.2810 & -0.0332 & -0.0002 \\ -0.0332 & 0.2026 & -0.0034 \\ -0.0002 & -0.0034 & 0.0001 \end{bmatrix}$$

Complete trajectory compared to the Monte-Carlo trajectory is shown in Fig. 21. Almost identical results are obtained in orientation and little shift exists in position.

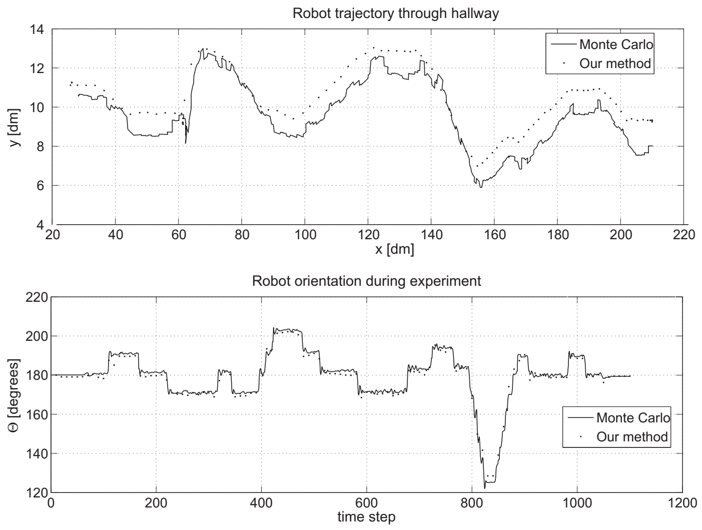


Fig. 21. Robot trajectory method comparison: solid - Monte Carlo localization, dots - our method

7. Conclusion

Monotonous position error growth is inherent characteristic of every mobile robot navigational system based solely on proprioceptive sensors. In order to deal with various sources of uncertainties in mobile robot localization it is necessary to establish a representative model of its internal states and environment and use perceptive sensors in the pose estimation. In this chapter we have demonstrated those properties on a differential drive mobile robot by localizing it in a 2D environment by using sonar ring as the perceptive sensor and in a 3D environment by using a mono camera as the perceptive sensor. In both cases we have applied nonlinear Kalman filtering for pose estimation and have compared results with the Monte Carlo localization based on a laser range finder, which is much more accurate sensor than sonars and cameras. Achieved localization accuracies with sonar ring and with mono camera are comparable to those obtained by the laser range finder and Monte Carlo localization. The applied calibration of mobile robot kinematic model also contributed to the increased accuracy.

8. References

*** (2000). *Optimization Toolbox For Use With Matlab User's Guide*, The MathWorks Inc.

*** (2007). ActivMedia robotics interface for application (ARIA): Overview, ActivMedia Robotics, LLC.

Aider, O. A., Hoppenot, P. & Colle, E. (2005). A model-based method for indoor mobile robot localization using monocular vision and straight-line correspondences, *Robotics and Autonomous Systems* **52**.

Arras, K. O., Tomatis, N., Jensen, B. & Siegwart, R. (2001). Multisensor on-the-fly localization: Precision and reliability for applications, *Robotics and Autonomous Systems* **34**.

Ashokaraj, I., Tsourdos, A., Silson, P. & White, B. (2004). Mobile robot localisation and navigation using multi-sensor fusion via interval analysis and ukf, *Proceedings of the 2004 Towards Autonomous Robotic Systems (TAROS), University of Essex,Colchester,UK* .

Borenstein, J., Everett, B. & Feng, L. (1996a). *Navigating Mobile Robots: Systems and Techniques.*, A. K. Peters, Ltd., Wellesley, MA, ISBN 1-56881-058-X.

Borenstein, J., Everett, H. R. & Feng, L. (1996b). *Where am I? Sensors and Methods for Mobile Robot Positioning*, University of Michigan, Ann Arbor, MI 48109.

Borenstein, J. & Feng, L. (1996). Measurement and correction of systematic odometry errors in mobile robots, *IEEE Transactions in Robotics and Automation* **12**(2).

Brezak, M., Petrović, I. & Ivanjko, E. (2008). Robust and accurate global vision system for real time tracking of multiple mobile robots, *Robotics and Autonomous Systems* **3**(56): 213–230.

Guilherme, N. D. & Avinash, C. K. (2002). Vision for mobile robot navigation: A survey, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**(2): 237–267.

Haykin, S. (2001). *Kalman Filtering and Neural Networks*, John Wiley and Sons, chapter Ch. 7. The Unscented Kalman Filter.

Ivanjko, E., Komšić, I. & Petrović, I. (2007). Simple off-line odometry calibration of differential drive mobile robots, *Proceedings of 16th International Workshop on Robotics in Alpe-Adria-Danube Region*, Ljubljana, Slovenia, pp. 164–169.

Ivanjko, E., Petrović, I. & Perić, N. (2003). An approach to odometry calibration of differential drive mobile robots, *Proc. International Conference on Electrical Drives and Power Electronics EDPE'03*, The High Tatras, Slovakia, pp. 519–523.

Ivanjko, E., Petrović, I. & Vašak, M. (2004). Sonar-based pose tracking of indoor mobile robots, *AUTOMATIKA - časopis za automatiku, mjerenje, elektroniku, računarstvo i komunikacije* **45**(3-4): 145–154.

Jeon, S. H. & Kim, B. K. (1999). Monocular-based position determination for indoor navigation of mobile robots, *IASTED Intl. Conf. on Control and Applications*, Banff, pp. 408–413.

Julier, S. J. & Uhlmann, J. K. (1996). A general method for approximating nonlinear transformations of probability distributions, *Technical report*, Dept. of Engineering Science, University of Oxford.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems, *Transactions of the ASME, Journal of Basic Engineering* **82**: 35–45.

Kälviäinen, H., Hirvonen, P., Xu, L. & Oja, E. (1994). Comparisons of Probabilistic and Non-probabilistic Hough Transforms, *Proc. of the 3rd European Conf. on Computer Vision*, Stockholm, Sweeden, pp. 351–360.

Konolige, K. (1999). Markov localization using correlation, *International Joint Conference on Artificial Intelligence*, Stockholm, Sweden.

Konolige, K. (2000). A gradient method for realtime robot control, *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)*, Kagawa University, Takamatsu, Japan.

Kosaka, A. & Kak, A. C. (1992). Fast Vision-Guided Mobile Robot Navigation Using Model-Based Reasoning and Prediction of Uncertainties, *CVIPG: Image Understanding* **56**(3): 271–329.

Lee, D. (1996). *The Map-Building and Exploration Strategies of a Simple Sonar-Equipped Robot*, Cambridge University Press, Trumpington Street, Cambridge CB2 1RP.

Li, X. J., So, A. & Tso, S. K. (2002). CAD-Vision-Range-Based Self-Localization for Mobile Robot Using One landmark, *Journal of Intelligent and Robotic Systems* **35**.

Moravec, H. P. & Elfes, A. (1985). High resolution maps from wide angle sonar, *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, St. Louis, USA, pp. 116–121.

Neira, J., Ribeiro, M. I. & Tardós, J. D. (1997). Mobile Robot Localization and Map Building using Monocular Vision, *5th Int. Symp. on Intelligent Robotic Systems*, Stockholm, Sweden, pp. 275–284.

Tsai, R. (1987). A versatile camera calibration technique for high accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses, *IEEE Journal of Robotics and Automation* **3**(4): 323–344.

Welch, G. & Bishop, G. (2000). An introduction to the Kalman filter, *Technical Report TR 95-041*, University of North Carolina at Chapel Hill, NC.

Wijk, O. (2001). *Triangulation Based Fusion of Sonar Data with Application in Mobile Robot Mapping and Localization*, PhD thesis, Royal Institute of Technology (KTH) Sweden, SE-100 44 Stockholm, Sweden.

www.blender3d.org (1995). *Blender Foundation* .

Global Localization based on a Rejection Differential Evolution Filter

M.L. Muñoz[1], L. Moreno[2], D. Blanco[2] and S. Garrido[2]
[1]*Facultad de Informática. Politechnical University of Madrid*
[2] *Robotics Lab. Carlos III University of Madrid*
Spain

1. Introduction

Autonomous systems are able to move from one point to another in a given environment because they can solve two basic problems: the localization problem and the navigation problem. The localization purpose is to determine the current pose of the autonomous robot or system and the navigation purpose is to find out a feasible path from the current pose to the goal point that avoids any obstacle present in the environment. Obviously, without a reliable localization system it is not possible to solve the navigation problem. Both problems are among the oldest problems in human travels and have motivated a considerable amount of technological advances in human history. They are also present in robot motion around the environment and have also motivated a considerable research effort to solve them in an efficient way.
The localization problem can be addressed in two main ways: on one hand, we have positioning systems and, on the other hand, we have self-localization systems. The *positioning systems* use external emitters (beacons) that are detected by on-board systems or an emitter located on board and several external receivers together with a communication system to send the robot the estimated pose. They use different variants of triangulation methods to estimate the robot pose at a given time. Different positioning systems can be found. The best known is the Global Positioning Systems (GPS) based on satellites around Earth and able to provide a localization in outdoor environments. For indoor environments the problem is more complex due to a high number of emitters and/or receivers required to obtain a complete coverage of the working area. Radio (Wifi and Zigbee), vision, and ultrasound-based systems are active research fields and have achieved an interesting development level, but these technologies depend strongly on the emitters and/or receivers distribution in the building. The positioning systems require to know the location of the emitters but they do not require to have an explicit map of the environment. Obviously, an implicit map is required at least to determine the distribution of the emitters or receivers along the environment. The global complexity of these indoor positioning systems is their weakest point, but it can be very interesting when the number of robots working in a given area is high. The *self-localization systems* use sensing systems located on board the vehicle and do not require any external system. Typical examples are ultrasound, laser, or vision-based localization systems where the emitter and the

receiver are located on the robot. This approach requires a map of the environment (predefined or learned) in order to estimate the robot pose. This chapter will focus on self-location systems, which are a little more autonomous than positioning systems, particularly in indoor environments linked to a mapping learning system.
The self-localization systems solve the pose estimation problem from two different initial situations:

- *Re-localization systems* (also called tracking systems): they try to keep tracking the mobile robot's pose, assuming the robot knows its initial position (at least approximately). Therefore, the self-localization system has to maintain the robot localized along the given mission. The majority of existing algorithms address only the re-localization problem because its mathematical treatment is less complex and, from a practical point of view, it is relatively simple to provide to the robot an initial pose. In this case, the small incremental errors produced along the robot motion and the initial knowledge of the robot's pose make classical approaches such as Kalman filters applicable. The Kalman filter for robot re-localization was introduced in the Eighties (Crowley, 1989; Cox, 1991; Leonard & Durrant-White, 1992; Jensfelt & Krinstensen, 1999) and has been extensively used. This type of filter constitutes a very efficient solution to the re-localization problem. However, the assumptions nature of the uncertainty representation makes Kalman filters not robust in global localization problems.
- *Global localization systems*: they do not assume any a priori knowledge about the robot's initial pose and therefore, they have to estimate the robot's pose globally. This problem has proven to be much more difficult to solve because the search space requires to use global techniques to explore or to integrate the received information until the pose converge to a unique solution.

From a mathematical point of view, the global localization problem can be solved using two different approaches: *Bayesian-based estimation methods* and *optimization-based methods*. In the first approach, *Bayesian methods* integrate all existent probabilistic information (sensor and motion information) into the posterior probability density at each motion-perception cycle, and the point estimate is posteriorly obtained as the state with bigger posterior probability density. Thus, these methods concentrate on the accurate modeling of the posterior probability density as a way to represent the most feasible hypothetical areas and their probabilities. At the convergence point the probability distribution is concentrated in a small area. This group of solutions has been extensively studied and the vast majority of current methods can be included here. Monte Carlo localization methods (Jensfelt et al., 2000) are purely Bayesian methods where the posterior probability distribution is modeled explicitly through the density obtained by the spatial distributions of particles (points with a given probability) along the search space. Other methods can be considered quasi-Bayesian, such as multi-hypotheses Kalman filters (Arras et al., 2002; Austin & Jensfelt, 2000; Jensfelt & Krinstensen, 1999; Cox & Leonard, 1994; Roumeliotis & Bekey, 2000), grid-based probabilistic filters (Fox et al., 1999; Burgard et al., 1996; Reuter, 2000) and, other hybrid solutions where the posterior probability distribution is modeled implicitly (Dellaert et al., 1999; Thrun et al., 2001). Multi-hypotheses Kalman filters are not completely Bayesian because, even if they maintain a set of multi-hypotheses, each of them with an associated Gaussian probability whose aggregated probability distribution can model the posterior, they do not operate on a pure Bayesian way since they use a decision tree search mechanism based on geometrical constraints together with probabilistic attributes to manage the global data association problem.

In the second approach, the *optimization-based methods* use also all existent probabilistic information to obtain a loss function that is minimized at each motion-perception cycle, and the point estimate is the point with lowest value of the loss function. Among the optimization-based approaches we can find differential evolution filters (Moreno et al., 2006) and particle swarm optimization filters (Vahdat et al., 2007). These groups of methods have been more recently used and, only in the last years, have been able to give efficient solutions to the problem. This chapter presents a solution to the global localization problem based on a modified version of the Evolutionary Localization Filter able to deal with the problems introduced by observation and motion noise in the optimization process. The method takes advantage of the capability of the Differential Evolution method to find the minima in complex global optimization problems by using a stochastic search approach.

2. Localization problem formulation

The robot's pose $(x, y, \theta)^T$ at time t will be denoted by x_t, and the data up to time t by Y_t. The posterior probability distribution according to this notation can be written as $p(x_t \mid Y_t, M)$, where M is the environment model which is known. To alleviate the notation, the term M is not included in the following expressions, $p(x_t \mid Y_t)$. The sensor data typically comes from two different sources: motion sensors which provide data related to change of the situation (e.g., odometer readings) and perception sensors which provide data related to environment (e.g., camera images, laser range scans, ultrasound measures). We refer to the former as motions u_i and to the latter as observations z_i. Motion $u(t-1)$ refers to the robot displacement in the time interval $[t-1, t]$ as a consequence of the control command given at time $t-1$. We will consider that both types of data arrives alternatively, $Y_t = \{z_0, u_0, \ldots, z_{t-1}, u_{t-1}, z_t\}$. These sensor data can be divided in two groups of data $Y_t \equiv \{Z_t, U_{t-1}\}$ where $Z_t = \{z_0, \ldots, z_t\}$ contains the perception sensor measurements and $U_{t-1} = \{u_0, \ldots, u_{t-1}\}$ contains the odometric information. To estimate the posterior distribution $p(x_t \mid Y_t)$, probabilistic approaches resort to the *Markov assumption*, which states that future states only depend of the knowledge of the current state and not on how the robot got there, that is, they are independent of past states.
From a Bayesian point of view, the global localization problem seeks to estimate the pose which maximizes the a posteriori probability density. This problem consists of two linked problems. On one hand, the integration of the probabilistic information available into the a posteriori probability density function of each state, given the set of motions, the set of measures and the a priori environment map of the environment. On the other hand an optimization problem to determine the point $\hat{x}_t^{MAP}$ with maximum a posteriori probability density at a given time.

$$\begin{aligned}
\hat{x}_t^{MAP} &= \arg\max_x p(x_t \mid Y_t) \\
&= \arg\max_x p(z_t \mid x_t, u_{t-1}, Y_{t-1}) p(x_t \mid x_{t-1}, u_{t-1}, Y_{t-1}) \\
&= \arg\max_x p(z_t \mid x_t) p(x_t \mid x_{t-1}, u_{t-1}) p(x_{t-1} \mid Y_{t-1}) \\
&= \arg\max_x \prod_{i=1}^{t} p(z_i \mid x_i) \prod_{i=1}^{t} p(x_i \mid x_{i-1}, u_{t-1}) p(x_0) \qquad (1)
\end{aligned}$$

This expression requires to specify $p(x_t | x_{t-1}, u_{t-1})$ and $p(z_t | x_t)$. Where $p(z_t | x_t)$ expresses the probability density function for the observation z_t, given the state x_t and an observation noise e, and $p(x_t | x_{t-1}, u_{t-1})$ indicates the probability density function for the motion noise ν. The expression (1) can be reformulated in an equivalent and more convenient form by taking logarithms:

$$\max_x [\sum_{i=1}^{t} \log p_e(z_i | x_i) + \sum_{i=1}^{t} \log p_v(x_i | x_{i-1}, u_{i-1}) + \log p(x_0)] \tag{2}$$

In general, the calculation of estimates for this optimization problem have no explicit analytical solutions for nonlinear and non-Gaussian models, and have to be iteratively solved to avoid the difficulties included in the optimization problem. These difficulties derive from the following aspects:

1. It is highly non-linear. Non-linearities due to motion and perception functions are propagated through the a posteriori robot pose probability density function.

2. Environment symmetries make the objective function to maximize multi-modal. At initial stages the objective function admits a high number of solutions, even with the same maximum value. That happens in highly symmetric environments, such as typical offices buildings. The reason can be noticed in (2), where the second term p_v is a constant in absence of robot's motion and the third term $p(x_0)$ is also constant in absence of initial pose information. This leads to an objective function $\max_x \sum_{i=1}^{t} \log p_e(z_i | x_i)$ which only depends on observations and has potentially multiple maxima in highly regular environments.

3. Another source of symmetries is originated by sensor limitations. The range and angular resolution of the sensor adds observational symmetries. Besides, some specific robot's poses can originate observational limitations which adds symmetries (e.g., a robot closes to a corner and looking at the corner).

In order to solve (2), a set of candidate estimates have to be initially generated, maintained or pruned according to the new observation and motion information included in the objective function. The problem is simplified in case the initial probability distribution is Gaussian, because the problem becomes uni-modal and then, it is possible to obtain, even analytically, an estimate (due to the problem can be converted into a quadratic minimization problem, if non linear motion and observation models can be approximated by a linear Taylor series expansion about the current estimate $\hat{x}_t$). This situation leads us to the well known Extended Kalman Filter solution of the position tracking problem.

We will use the notation $f_0(x)$ to refer the objective function to maximize. The problem of finding an x that maximizes $f_0(x)$ among all x that satisfy the conditions $x_{t+1} = f(x_t, u_t) + v_t$ and $z_t = h(x_t) + e_t$ is limited to finding the optimal value within the set of all feasible points. A pose is feasible if it satisfies the constraints $f()$ and $h()$. In the problem under consideration, there exist, at least at initial stages, multiple optimal values. Thus, the methods to solve the problem require to be able to manage a set of solutions. The Bayesian methods use the a posteriori probability density function to do that, as was previously commented. The method proposed here uses a different approach. The idea is to maintain a set of feasible solutions to the localization problem, and let this set evolve towards optimal values according to the observed motion and perception data.

2.1 Recursive formulation

The MAP estimate formulated as an optimization problem subject to conditions, in equation (2), is not practical from a computational point of view. To implement a global localization algorithm in a robot, a recursive formulation is required. The objective function $f_0(x_t)$ can be expressed recursively in the following way:

$$\begin{aligned}
f_0(x_t) &= \sum_{i=1}^{t} \log p_e(z_i \mid x_i) + \sum_{i=1}^{t} \log p_v(x_i \mid x_{i-1}, u_t) + \log p(x_0) \\
&= \log p_e(z_t \mid x_t) + \sum_{i=1}^{t-1} \log p_e(z_i \mid x_i) \\
&+ \log p_v(x_t \mid x_{t-1}, u_{t-1}, m) + \sum_{i=1}^{t-1} \log p_v(x_i \mid x_{i-1}, u_{t-1}) + \log p(x_0) \\
&= \log p_e(z_t \mid x_t) + \log p_v(x_t \mid x_{t-1}, u_{t-1}) + f_0(x_{t-1})
\end{aligned} \tag{3}$$

If we are able to solve the optimization problem at time $t-1$, and we have a set of sub-optimal solutions which satisfy the optimization problem up to time $t-1$, the MAP optimization problem can be reformulated as

$$\hat{x}_{t-1} = \max_x \log p_v(z_t \mid x_t) + \log p_e(x_t \mid x_{t-1}, u_{t-1}) \tag{4}$$

where $\hat{x}_{t-1}$ is the x which maximize the MAP optimization problem at time $t-1$, and x^*_{t-1} is the population set of sub-optimal solutions at the end of iteration $t-1$. Then solving (4) we will obtain a recursive version of the MAP estimate.

3. Evolutionary Localization Filter algorithm

3.1 Differential Evolution: basic concepts

The algorithm proposed to implement the adaptive evolutive filter is based on the differential evolution method proposed by Storn and Price (Storn & Price, 1995) for global optimization problems over continuous spaces. The Adaptive Evolutionary Localization Filter uses as a basic solution search method, the classical **DE/rand/1/bin** version with some modifications to improve its characteristics in presence of a noisy fitness function.

The **DE/rand/1/bin** uses a parallel direct search method which utilizes n dimensional parameter vectors $x_i^k = (x_{i,1}^k, \ldots, x_{i,n}^k)^T$ to point each candidate solution i to the optimization problem at iteration k for a given time step t. This method utilizes N parameter vectors $\{x_i^k; i = 1, \ldots, N\}$ as a sub-optimal feasible solutions set (population) for each generation t of the optimization process.

The initial population is chosen randomly to cover the entire parameter space uniformly. In absence of a priori information, the entire parameter space has the same probability of containing the optimum parameter vector, and a uniform probability distribution is assumed. The differential evolution filter generates new parameter vectors by adding the weighted difference vector between two population members to a third member. If the resulting vector yields a lower objective function value than a predetermined population member, the newly generated vector replaces the vector with which it was compared; otherwise, the old vector is retained. This basic idea is extended by perturbing an existing vector through the addition of one or more weighted difference vectors to it (see fig. 2).

3.1.1 Differential Perturbation Operation

The perturbation scheme generates a variation v_i^k according to the following expression,

$$v_i^k = x_{r_1}^k + F(x_{r_2}^k - x_{r_3}^k) \tag{5}$$

where $x_{r_1}^k$, $x_{r_2}^k$ and $x_{r_3}^k$ are parameter vectors chosen randomly from the population, different from running index i, and mutually different. F is a real constant factor which controls the amplification of the differential variation $(x_{r_2}^k - x_{r_3}^k)$.

3.1.2 Crossover Operation

In order to increase the diversity of the new generation of parameter vectors, a crossover is introduced. The new parameter vector is denoted by $u_i^k = (u_{i,1}^k, u_{i,2}^k, \ldots, u_{i,n}^k)^T$ with

$$u_{i,j}^k = \begin{cases} v_{i,j}^k & \text{if } p_{i,j}^k < \epsilon \\ x_{i,j}^k & \text{otherwise} \end{cases} \tag{6}$$

where $p_{i,j}^k$ is a randomly chosen value from the interval (0,1) for each parameter j of the population member i at step k, and ε is the crossover probability and constitutes the crossover control variable. The random values $p_{i,j}^k$ are made anew for each trial vector i.

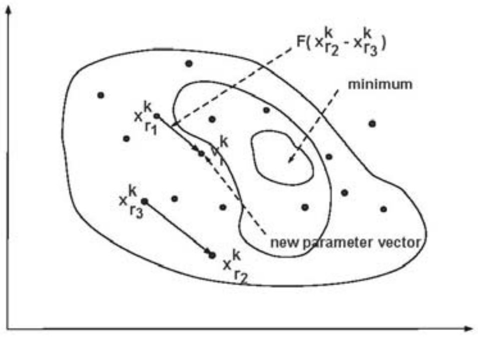


Fig. 1. New population member generation.

3.1.3 Selection Operation

To decide whether or not vector u_i^k should become a member of generation $i+1$, the new vector is compared to x_i^k. If vector u_i^k yields a value for the objective fitness function better than x_i^k, then is replaced by u_i^k for the new generation; otherwise, the old value x_i^k is retained for the new generation.

3.1.4 Shift Operation

After the DE algorithm has completed its iterations, the points included in the population set x_t^* are moved according to the robot motion model $x_{t+1}^i = f(x_t^i, u_t)$, the candidate pose and the observed odometric data.

3.1.5 Fitness function

According to the optimization problem under consideration, $\max_x(\log p_v(z_t \mid x_t) + \log p_e(x_t \mid x_{t-1}, u_{t-1}))$, the natural choice for fitness function is

$$f_0(x^t) = \log p(z_t \mid x_t) + \log p(x_t \mid x_{t-1}, u_{t-1}) \tag{7}$$

This expression contains two probability densities associated to errors in the motion and observation models (the perception error probability density distribution $p(z_t \mid x_t)$ and the robot's motion error probability density distribution $p(x_t \mid x_{t-1}, u_{t-1})$). A third probability model is used to model the information we have at initial stage about the initial a priori robot's pose $p(x_0)$. This initial probability pose distribution is used at the initial phase to distribute the population set of the ELF algorithm. In case of global localization problem, the initial pose information is null. Then, the population set is distributed according to a uniform probability distribution along the space state.

To compute $p(z_t \mid x_t)$, it is necessary to predict the value to be observed by the sensor, assuming that the robot's pose estimate is known. Let assume the pose estimate is $\hat{x}_t$, the sensor relative angle with respect to the robot axis is α_i and a given environment model m. According to the observation model, the noise-free predicted measurement will be $\hat{z}_{t,i} = h(\hat{x}_t, \alpha_i, m)$ (in our case, $\hat{z}_{t,i}$ is computed using a ray tracing method). Assuming that the measurement error is Gaussian with zero mean and known covariance ($e_{t,i} \approx N(0, \sigma_e)$), the predicted measurement will be the center of the Gaussian probability distribution of the expected distance measured by the α_i sensor when robot is located at x_t. Then the probability of observing $z_{t,i}$ with sensor i can be expressed as

$$p(z_{t,i} \mid \hat{x}_t) = \frac{1}{(2\pi\sigma_e^2)^{1/2}} e^{-1/2\frac{(z_{t,i}-\hat{z}_{t,i})^2}{\sigma_e^2}} \tag{8}$$

The integration of all individual sensor beam probabilities into a joint probability value, assuming conditional independence between the individual measurements, is expressed as

$$p(z_t \mid \hat{x}_t) = \prod_{i=0}^{N_s} p(z_{t,i} \mid \hat{x}_t) = \prod_{i=0}^{N_s} \frac{1}{(2\pi\sigma_e^2)^{1/2}} e^{-1/2\frac{(z_{t,i}-\hat{z}_{t,i})^2}{\sigma_e^2}} \tag{9}$$

where N_s is the number of sensor observations.

The second probability required to calculate the objective function is $p(x_i \mid x_{i-1}, u_{t-1})$. To compute $p(x_i \mid x_{i-1}, u_{t-1})$, we have to predict the robot's pose $\hat{x}_t$ assuming that the robot's pose estimate is $\hat{x}_{t-1}$ and taking into account the motion command u_{t-1} at that cycle. Let $\hat{x}_t = f(\hat{x}_{t-1}, u_{t-1})$ denote this ideal predicted state. Assuming the motion error is a zero mean with known variance Gaussian probability distribution (that is $\nu \approx N(0,P)$), this predicted measure will be the center of the Gaussian probability distribution of the expected distance when the robot is located at $\hat{x}_t$. Then, the $p(x_i \mid x_{i-1}, u_{t-1})$ probability can be expressed as

$$p(x_i \mid x_{i-1}, u_{t-1}) = \frac{1}{\sqrt{|P|(2\pi)^n}} e^{-1/2(x_i - \hat{x}_i)P^{-1}(x_i - \hat{x}_i)^T} \tag{10}$$

Introducing the expressions of $p(x_t \mid x_{t-1}, u_{t-1})$ and $p(z_t \mid x_t)$ in (7)

$$\begin{aligned} f_0(x_t) &= \log \prod_{i=0}^{N_s} (2\pi\sigma_e^2)^{-1/2} e^{-\frac{(z_{t,i} - \hat{z}_{t,i})^2}{2\sigma_e^2}} \\ &+ \log(|P|(2\pi)^n)^{-1/2} e^{-\frac{1}{2}(x_i - \hat{x}_i)P^{-1}(x_i - \hat{x}_i)^T} \\ &= \sum_{i=0}^{N_s} \log(2\pi\sigma_e^2)^{-1/2} - \sum_{i=0}^{N_s} \frac{(z_{t,i} - \hat{z}_{t,i})^2}{2\sigma_e^2} \\ &+ \log[(|P|(2\pi)^n)^{-1/2}] - \frac{1}{2}(x_i - \hat{x}_i)P^{-1}(x_i - \hat{x}_i)^T \end{aligned} \tag{11}$$

which can be reduced to find the robot's pose to minimize the following function

$$f'_0(x_t) = \sum_{i=0}^{N_s} \frac{(z_{t,i} - \hat{z}_{t,i})^2}{2\sigma_e^2} + \frac{1}{2}(x_i - \hat{x}_i)P^{-1}(x_i - \hat{x}_i)^T \tag{12}$$

The differential evolutive localization filter will minimize iteratively the fitness function (12) and then, the displacement is evaluated according to the odometric information. The objective fitness function let us notice that the optimization considers the quadratic observation error and the quadratic pose error between the predicted pose and the pose under consideration weighted according its covariance matrix.

4. Difficulties to achieve a robust method

Different problems need to be solved in optimization methods to achieve a reasonable robustness level. Some of this problems are general and related to the global localization problem, and others are related to the nature of the basic algorithm adopted. Among the general problems, we can remark the following ones:

• The lack of a method to determine the significance of a new point apart from the fitness function value. This originates two classes of problems: **premature convergence** to a local minima and, in case of multiple hypotheses situations, the premature elimination of feasible hypotheses. Both situations originate a fail in the convergence to the true global pose. This second situation can be observed in figure 2, where the red points are the pose hypotheses at population set. In this case, the robot is localized at an office exactly equal to others in dimensions and furniture. Due to the fact that the robot's orientation is 270 degrees, it can not distinguish between offices and the hypothesis should be maintained trough iterations. But, if we let the algorithm iterate, it can be observed how some of the hypotheses are removed. This process can end in one hypothesis. Obviously, this algorithm behavior is not robust.

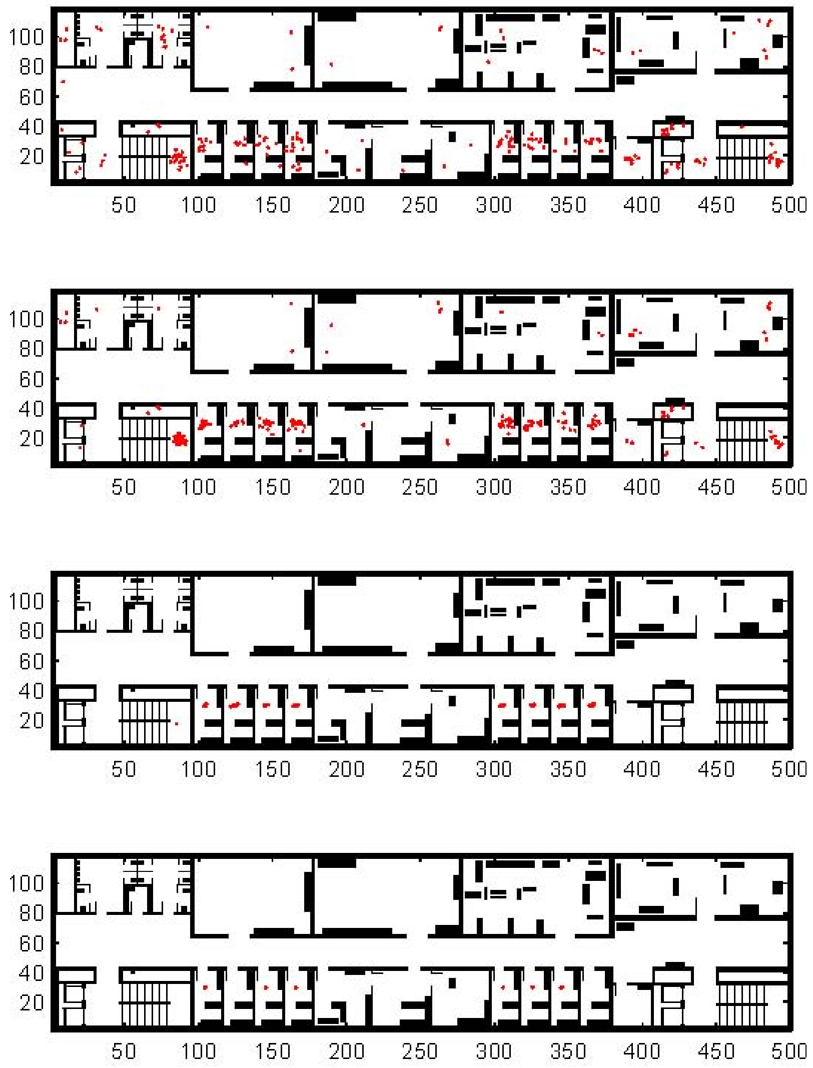


Fig. 2. Premature hypothesis elimination, initial pose (105, 30, 270), 200 elements in population and σ of 3% of the measured signal.

A traditional solution is to increase the population number to make more difficult the premature elimination of feasible hypotheses and to limit the number of iterations to control the algorithm progress.

- A second important problem is how to determine **the stopping criteria**. Traditional methods are: to fix a predefined iteration number or to stop the algorithm when the fitness function does not improve for a number of iterations. However the use of a predefined iteration number can also lead to premature elimination of hypotheses, as can be observed in the example of figure 3, where, after three motions the algorithm, it has converged prematurely to an incorrect pose. In the figure, the blue cross indicates the best fitness pose of the population set and the magenta points show the observed laser measurements projected over the environment map according to the best pose estimate.

The second idea for the stopping criteria consists on stopping the algorithm if after a predefined number of iterations the best hypothesis is not modified. This stopping criteria is not easy to establish. If the number of iterations without improvement is low, the algorithm can not converge or converge very slowly. This problem can be observed in the example of figure 4 that shows a stopping criteria of 20 cycles without best fitness improvement. In the figure can be notice that the algorithm can not converge properly. This problem can be partially eliminated by using a bigger number of default cycles. In that case, the algorithm does more iterations at each perception-motion cycle but then, we move to the previous case situations where the algorithm converges prematurely to an improper pose. Obviously, we can try to adjust the parameter to each situation, but this adjust depends on the observation and motion noises, on the shape and size of the environment observed at initial pose and, consequently, it is not robust.

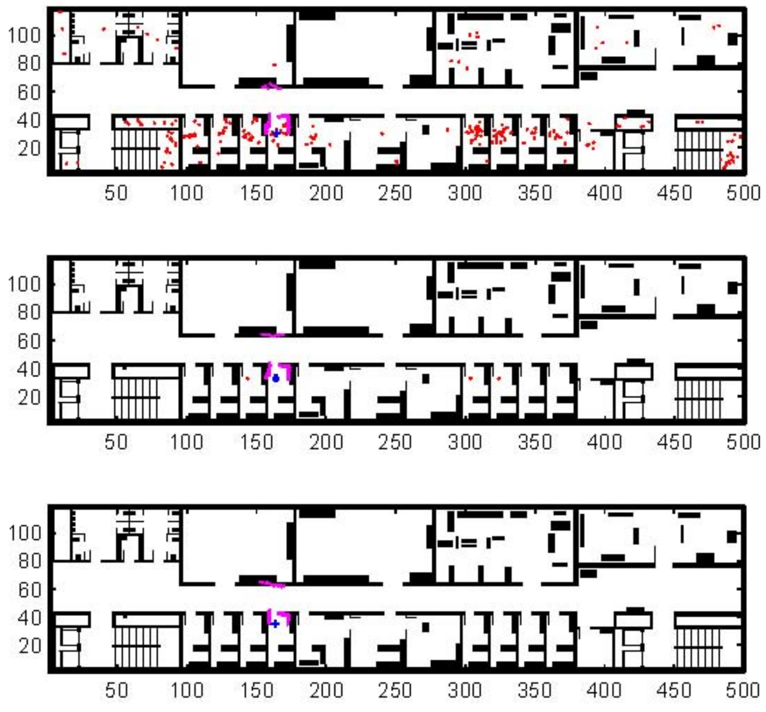


Fig. 3. Premature hypothesis elimination, starting pose (305, 30, 270), 200 elements in population and σ of 3% of the measured signal, motion +2.5 cells per cycle in y direction.

• A third problem is the population dispersion. The problem can be perceived in the last image of figure 4, where the population set is dispersed considerably when the robot goes out of the office and passes to the corridor. Since Differential Evolution is a stochastic search method, the pose set spreads along the best fitness areas contained in the stochastic search ball (defined by the possible combinations of three elements stochastically taken from the population set). If the population has not converged, it spreads when the robot moves to an area where many possible poses has a similar fitness value. This problem is originated by the lack of memory of each individual pose of the population set in the algorithm.

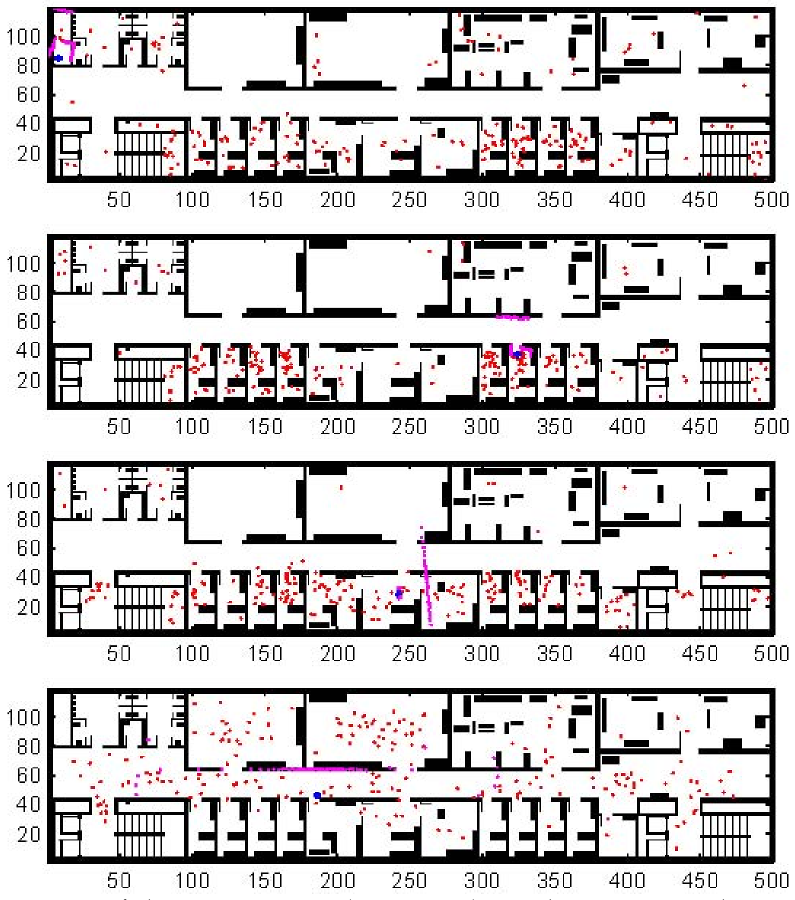


Fig. 4. Convergence failure, starting pose (305, 30, 270), 200 elements in population and σ of 1% of the measured signal, motion +2.5 cells per cycle in y direction.

4.1 Solutions to deal with noisy fitness function problems

The two first problems are originated when a superior candidate solution may be erroneously considered as an inferior solution due to the noise and eliminated from the set of solutions by the algorithm. Different solutions has been proposed in literature to compensate the noise problem in evolutive algorithms:

• Increasing the population size. It is the most simple way to deal with the noise problem. This technique reduces the probability of eliminating correct hypothesis prematurely, but increases the computational cost of the algorithm. This solution does not solve the dispersion problem.

• Resampling and averaging the fitness function from several samples reduces the error in the fitness. This method assumes as estimate to evaluate the fitness function the sampling mean that has its standard error reduced by $\sqrt{n}$, where n is the number of samples used to estimate the mean. This technique is frequently used by the optimization researchers, but it requires to sample the fitness function repeatedly. This technique can not be used for dynamical systems because these systems do not remain in the same state and consequently, they can not be used for the global localization system problem.

• Thresholding was proposed by Markon (Markon et al., 2001). The idea is to replace an existing candidate solution only when the fitness difference is larger than a given threshold τ. This method requires to calculate the threshold value, which depends on the variance of the noise and the fitness distance to optimal fitness value. This mechanism requires to increase the number of iterations, since the level of candidate solutions rejected increases.

• Estimation of the true fitness. This idea was suggested by Branke (Branke et al., 2001). He proposes to estimate an individual fitness using a local regression of the fitness of the neighboring individuals. The underlying assumptions of this method are: that the true fitness function can be locally approximated by a low polynomial function, that the variance in a local neighborhood is constant, and the noise is normally distributed.

The third problem (dispersion) requires a different approach and has not be widely studied in literature (perhaps because it appears mostly in dynamical system subject to noise).

5. Rejection Differential Evolution Filter

The solution adopted in this work use three main mechanisms to improve the robustness and efficiency of the basic DE algorithm to solve the global localization problem. These mechanisms are:

1. A **threshold rejection band** to avoid the premature elimination of solutions. This mechanism decreases the eagerness of the algorithm, allowing it to eliminate a candidate solution from the set only when the offspring candidate is significatively better from a statistical point of view.

2. An **stopping criteria** based on the expected fitness value to stop the algorithm iterations in a statistically equivalent point. This idea will let the algorithm iterate as much as possible to obtain a fast convergence towards the solution if there is a statistical improvement between iterations or to stop very fast if no statistical improvement is obtained.

3. Adjustment of the **perturbation amplification factor** F. This mechanism tries to maintain a high amplification factor while the population evolves in the first perception cycle to the most the promising areas (a wide scope search is required) and then to limit the algorithm search scope when the population set is distributed in the most feasible areas.

5.1 Threshold determination

The fitness function to minimize is given by expression (12). For a given candidate x_t^j the fitness function value is given by

$$f_0^j(x_t^j) = \sum_{i=0}^{N_s} \frac{(z_{t,i} - \hat{z}_{t,i}^j)^2}{2\sigma_e^2} + \frac{1}{2}(x_t^j - \hat{x}_t)P^{-1}(x_t^j - \hat{x}_t)^T \tag{13}$$

where $z_{t,i}$ is the measure given by the range scan sensor at angle α_i at cycle t, $\hat{z}_{t,i}^j$ is the estimated observation for the candidate robot's pose x_t^j, and x_t is the pose estimate (if it exists at cycle t). The second term of the expression depends on the robot pose estimate $\hat{x}_t$ that is not known at initial step, and it is neglected until a unique pose estimate is obtained (that happens when all population has converged to a limited area around the best pose estimate). The fitness function before the convergence point information takes the following form:

$$f_0^j(x_t^j) = \sum_{i=0}^{N_s} \frac{(z_{t,i} - \hat{z}_{t,i}^j)^2}{2\sigma_e^2} = \frac{1}{2}\sum_{i=0}^{N_s} \frac{v_{t,i}^2}{\sigma_e^2} \tag{14}$$

where $v_{t,i} = (z_{t,i} - \hat{z}_{t,i}^j)$ represents the discrepancy between the observed and the predicted value of the sensor data. To estimate the expected noise band for the fitness function, we need to calculate the expected value for $E[f_0]$ when the pose under evaluation is the true one. The term $\sum_{i=0}^{N_s} v_{t,i}^2/\sigma_e^2$ where $v_{t,i}/\sigma_e^2$ are standard normal random variables $N(0,1)$ is a chi-square distribution with N_s degrees of freedom. This distribution is well known and it has mean N_s and variance $2N_s$. Then, the expected minimum fitness value will be

$$E[f_0] = \int_{-\infty}^{+\infty} f_0(v)p(v)dv = N_s/2 \tag{15}$$

That means that, even if the pose we are considering was the true robot's pose, the expected fitness function value would be $N_s/2$ due to observation errors produced at the perception time. If two candidate poses x_1 and x'_1 are compared at a given iteration time, the question is: when can we consider there exists a reasonably evidence that candidate pose x_1 is better than x'_1 ?. In the tests, different values for the threshold rejection level have been simulated. To maintain the elitism in the method, one exception has been introduced (a pose candidate with a fitness better than the best pose existent up to that moment will always pass to the following iteration). That exception consists of selecting the best pose obtained for the next iteration population, independently of the rejection threshold.

5.2 Stopping condition

A classical problem in optimization methods is how to determine a stopping condition. This problem can be considered in different ways: limiting the number of iterations, iterating until a pre-specified accuracy is obtained or iterating until no additional improvement is obtained. But in case of noisy fitness problems, those conditions are not appropriate.

Assuming that the fitness function is a chi-square with N_s degrees of freedom, it is possible to obtain the p-quantile function value with a pre-specified p value of probability, or in other words, the fitness function value f_{1-p} that has $1-p$ probability of being inferior to the fitness function value at any perception cycle. Quantile values for some pre-specified probability values and degrees of freedom can be found in statistics literature.

5.3 Amplification factor

The previous mechanisms improve greatly the robustness but they do not exploit the local convergence. This effect is clearly evident in office buildings where many offices have the same dimensions, which originates a multiple convergence areas. Once the robot gets out of the office to a corridor, if factor F is maintained high, the population spreads along the corridor areas.

To avoid this problem, the amplification factor is initialized at $F = 0.99$ in the first iteration of the observation cycle and, after the first perception cycle, F is decreased to a low value, $F = 0.05$. This tends to keep the search area of the algorithm at initial perception cycle as wide as possible and, once the algorithm has localized the most feasible areas, the amplification factor is decreased to a low value to concentrate the exploration in the surroundings of the previous areas avoiding an unnecessary dispersion of the population.

6. Convergence results

To test the algorithm characteristics, a simulated environment has been considered (figure 5). This environment is similar to many office indoor areas. All offices are localized along the central corridor. The offices localized on the upper part of the figure have the same length in y dimension and an x length progressively decreasing (in one cell) from offices localized on the left side of the figure to those located on the right side. On the contrary, offices localized on the lower part of the figure are of exactly the same dimensions and appearance. The offices localized on the upper and lower corners of the environment have similar dimensions but doors are localized on different sides.

6.1 Test 1

The first test tries to determine the capability of the algorithm to localize the robot when it is localized at a distinguishable pose. The true robot's position is $(x, y, \theta)^T = (60, 60, 0)^T$ and the variance considered for each measurement in the simulation is of 3% of the measured signal, which is relatively noisy compared with real laser scanners. The population set used in the test is of 100 elements.

In the test example of figure 5, the stopping condition happens at iteration 334. At that point the estimated pose is $(60.232, 60.098, 359.839)$ (units are in cells and degrees). The size of the cell considered for the map is of 12 cm, which corresponds to an estimation error of 2.79 cm in x dimension, 1.182 cm in y and 0.16 degrees in orientation, which is quite accurate for the noise level and for one perception cycle. In figure 5, the red points indicate the candidate poses position, the magenta points represent the points observed in the environment according to the best pose obtained, and the blue cross represents the best pose obtained. If we increase the noise level, the number of feasible points at the end of the first perception cycle tends to increase, since to the noise level tends to make the disambiguation capability of the method more difficult.

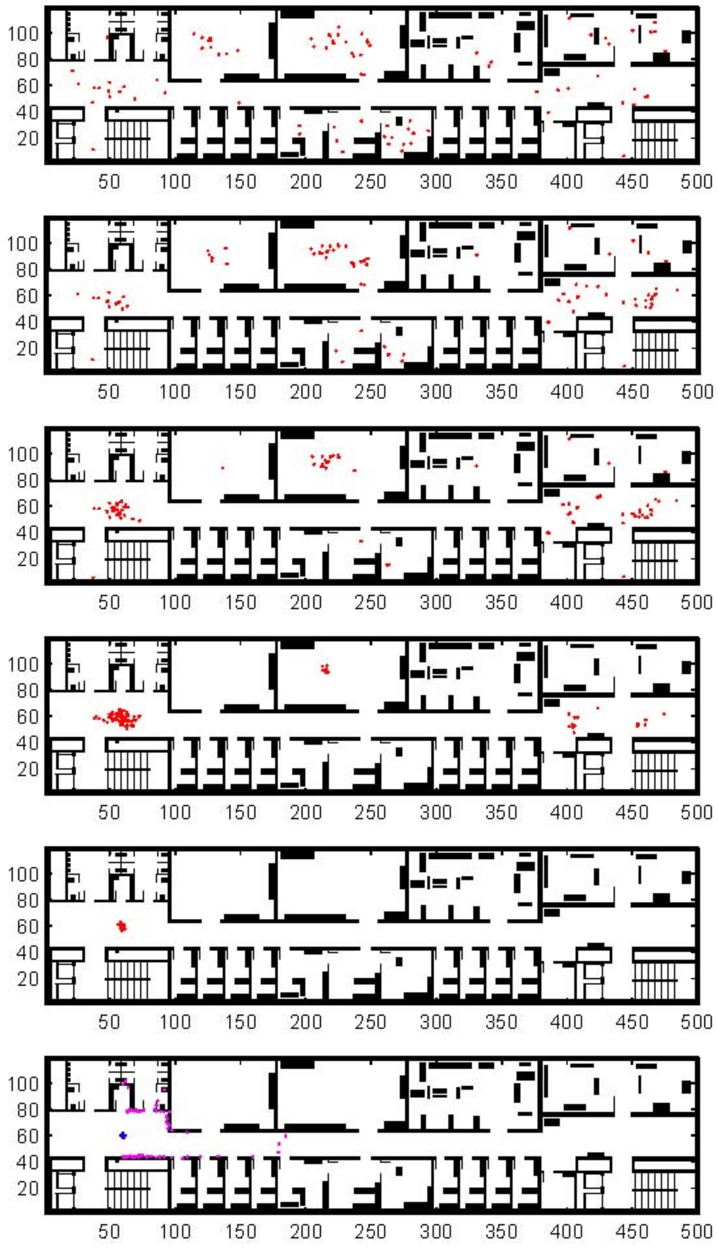


Fig. 5. Convergence process along different iterations (50, 100, 150, 200, 250, 334**)** of the first perception cycle.

6.2. Test 2

The second test tries to determine the capability of the algorithm to identify all possible feasible areas in case the robot is located in a non-distinguishable pose. The true robot position is $(x, y, \theta)^T = (60, 60, 0)^T$ and the variance considered for each measurement in the simulation is of 3% of the measured signal, which is relatively noisy compared with real laser scanners. The population set used in the test is of 100 elements. Figure 6 shows the convergence population process along the iterations of the first perception cycle. It can be noticed how the algorithm is able to concentrate the population in the most feasible areas (in this case, in offices not localized in front of a laboratory door). The most interesting aspect of this second test is that the algorithm does not eliminate any potential hypothesis.
After the first perception cycle the robot is moved upward (y direction) in the map: 2.5 cells at each motion plus a Gaussian error. After the motion, the whole population set is moved according to the odometry information and a new localization cycle is started, this time with an amplification factor $F = 0.05$. Figure 7 shows the successive population convergence toward an only hypothesis.
In the test example of figure 7, the stopping condition changes following the next sequence: 314, 24, 13, 11, 68 and 9. It can be noticed how the algorithm is heavier in the first perception cycle since it needs to eliminate infeasible areas which require a high number of pose trails. After that iteration, the stopping criteria is reached faster, requiring a number of iterations of two orders of magnitude to converge. It can also be noticed that the number of iterations increases when the robot goes out of the office and perceives the corridor. In that case, the algorithm requires 68 iterations before reaching the stopping criteria. Once the robot observes the corridor, it is able to converge to only one feasible pose area, since the observations that let the algorithm disambiguate between the offices.

6.3. Test 3

The third test tries to determine the capability of the algorithm to identify all possible feasible areas in case the robot is localized at the worst possible case. The worst case happens when the robot is localized at a corner and observes the corner from a short distance. In that case, it is a non-distinguishable pose and the number of possible feasible poses exploits. The true robot's position is $(x, y, \theta)^T = (10, 70, 135)^T$, which corresponds to the upper left corner of the hall localized at the left side of the test environment. The variance considered for each measurement in the simulation is of 1% of the measured signal. Due to the fact that the number of potential feasible poses is high, if a normal population is used (we understand by normal population a population able to localize the vehicle in normal situations), the algorithm fails because it does not have enough elements to manage the set of potential hypotheses. In our experimental test, a minimum number of approximately $15 - 25$ elements per potential pose is required. A population set of 1500 of elements has been used to manage properly the high number of feasible hypotheses existent at the initial cycle of the global localization. Figure 8 shows the high number of feasible poses existent at the initial robot's pose according to the perceived information. The number of feasible poses at the end of the first perception cycle is of 54. In this example, a population of is enough, but at the end of the first cycle it has feasible poses. If we decrease the population, the number of manageable poses decreases and the risk of incorrect convergence increases.

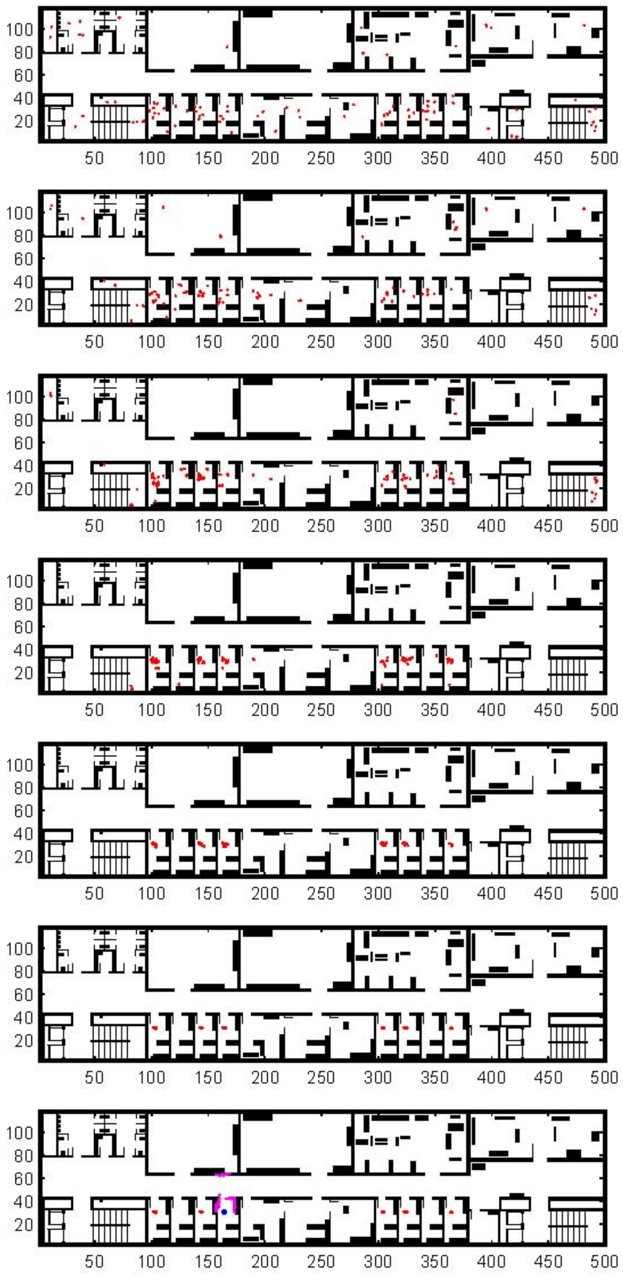


Fig. 6. Convergence process along different iterations (50, 100, 150, 200, 250, 300, 314**)** of the first perception cycle.

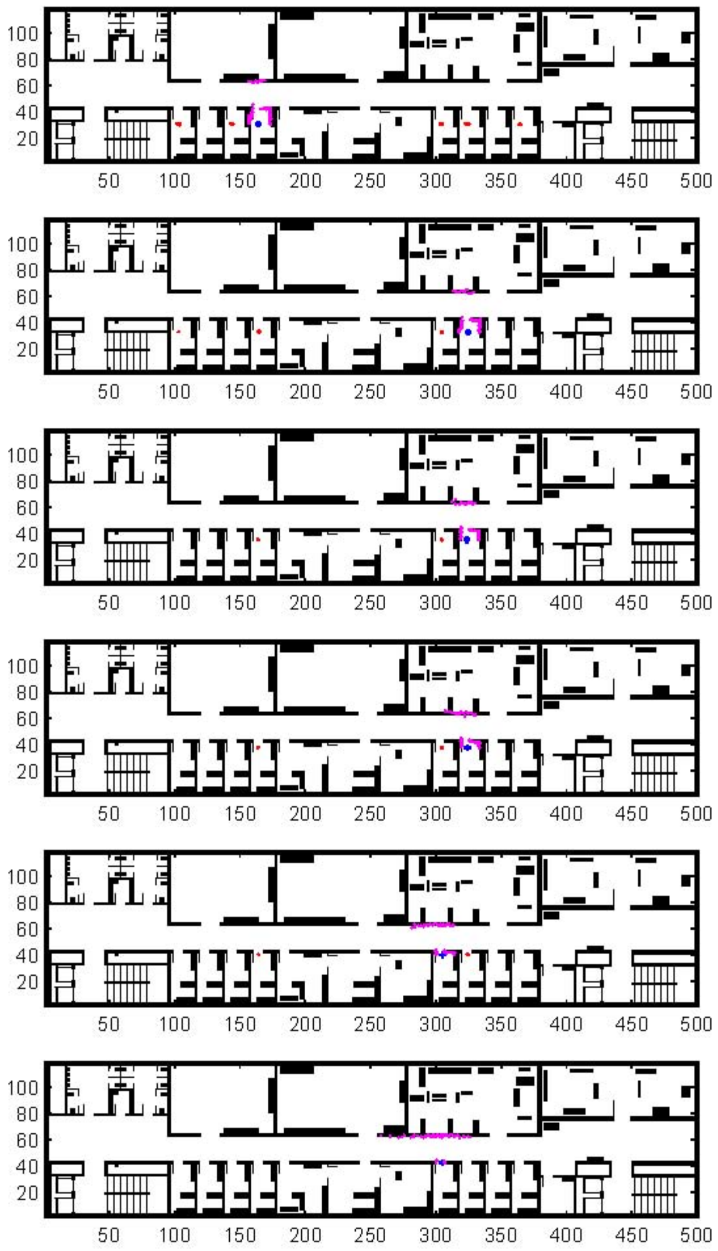


Fig. 7. Convergence process along 6 successive perception-motion cycles where the robot moves 2.5 cells upward after each perception cycle.

After the first perception cycle, the robot is turned 10 degrees clockwise at each motion (plus a random error added and unknown for the algorithm). After the motion, the whole population set is moved according to the odometry information and a new localization cycle is started. Figure 9 shows the successive population convergence towards an unique hypothesis. After the initial perception cycle, the number of possible poses is pruned very fast since new information about the environment is added to the algorithm.

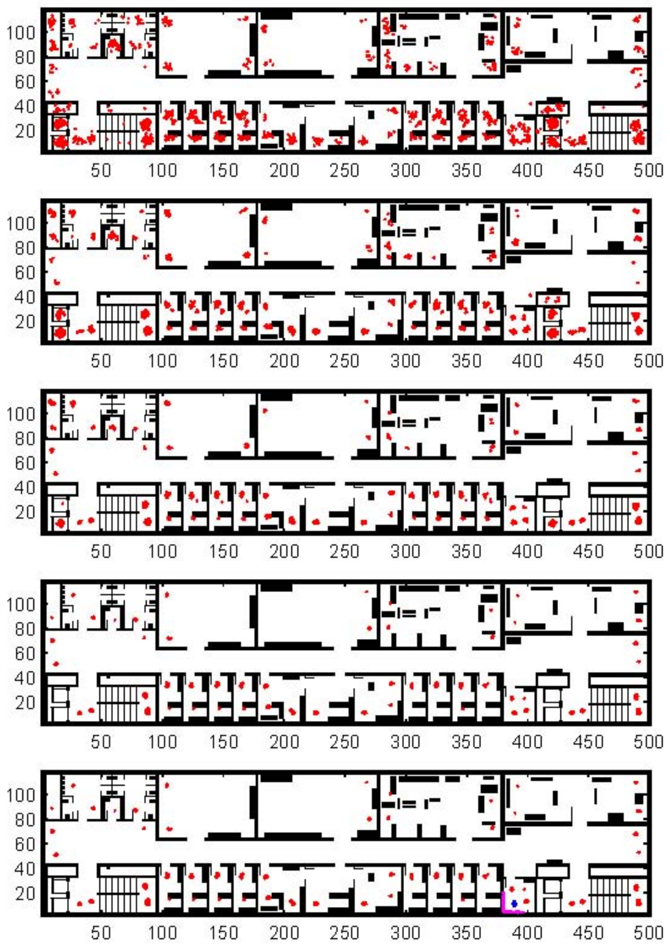


Fig. 8. Convergence process for the worst case pose (100, 200, 300, 400, 420) of the first perception cycle.

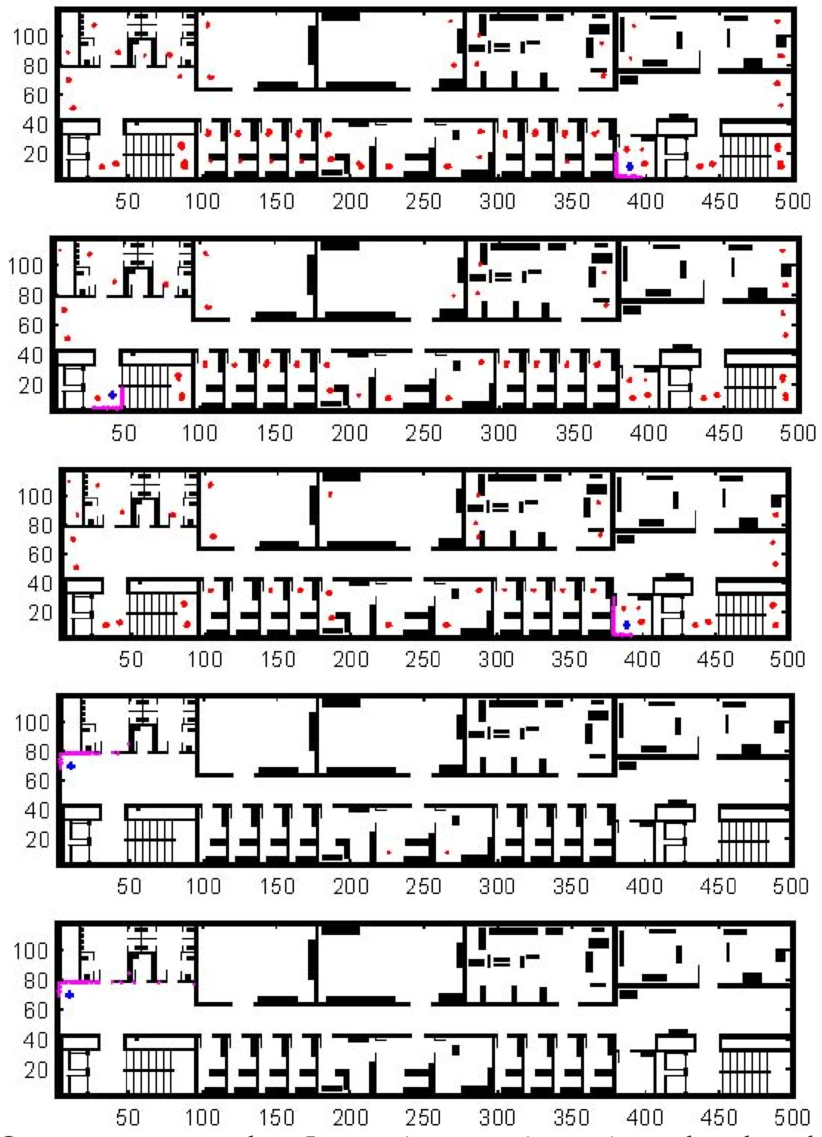


Fig. 9. Convergence process along 5 successive perception-motion cycles where the robot turns 10° clockwise after each perception cycle.

From a practical point of view, the worse case situation is easy to alleviate by turning the vehicle up to a pose with the maximum perception information. For this same example, if we start the localization with an initial pose of $(x, y, \theta)^T = (10, 70, 135)^T$ where the perceived area covered by robot sensors is maximum, the problem disappears and a population of 100 elements is enough, as can be noticed in figure 10.

7. Accuracy results

To test the accuracy of the method for the initial localization, we have selected again a distinguishable point (x,y,θ)=(60,60,0) and we have increased the variance of the error in sensor measurement. The variance considered in simulations is proportional to the measured signal and is expressed as a fraction of the total measurement. This way, a value of 0.01 indicates a variance of 1 % over the simulated value obtained for a given sensor. This situation is harder than real conditions in laser scanners. The population used in simulation is of 100 elements. For each case, 25 runs of the first perception cycle have been executed. The results are shown in table 1, where the mean and variance of the absolute errors in x, y and θ are given. It also shows the average number of iterations required by the algorithm until the stopping criteria is reached and the success ratio obtained for each noise level.

It can be notice that, for low noise variance levels (up to 5%), the accuracy of the algorithm is below 0.15 cells in x and y dimensions and below 0.2° in orientation in all the cases. Since the cell size used is 12 cm, that means an error below 1.8 cm in x and y dimensions and below 0.2° in orientation at the end of the first perception cycle. For this signal error level, the algorithm has successfully localized the true pose in all the runs and only one hypothesis is maintained at the end of the first perception cycle. The stopping criteria is reached in a relatively constant number of iterations for low noise levels, and it tends to decrease slowly when the noise signal level increases.

The algorithm degrades slowly. For a 17.5% of variance in the noise level, the algorithm is able to localize a position close to the true one in all simulations. We consider a localization as successful if the initial pose estimated is in a 10 cells area around the true one. After that level, the success ratio drop fast and, for a 25% of variance in the noise level, the success ratio value is only of a 60%.

From this test, some conclusions can be drawn. The first one is that, if the place is distinguishable, the algorithm is able to localize the initial pose with high accuracy, and only when the noise increases considerably the algorithm starts to decrease its success ratio.

A second aspect to consider is the capability of the algorithm to maintain bounded the pose estimation accuracy along a trajectory. A motion along the central corridor in the test environment is simulated. For the simulations, a normal error with 2.5% of variance has been adopted and the noise used for sensor observation is of 1% of variance. The real and estimated trajectories are shown in figure 11 and the x, y, and θ errors are shown in figure 12. The simulation results show that y error is very small and contained between [+0.05, -0.05] cells ([+0.6, -0.6] cm). This is logical, since the corridor is relatively narrow and the y position can be accurately estimated with the available information. For x variable, errors are in a band between [+0.1, -0.1] cells ([+1.2 -1.2] cm) and they punctually reach values in the band [+0.25, -0.25] cells. Regarding the orientation, θ errors are in a band between [+0.1, -0.1]° and they punctually reach values in the band [+0.25, -0.25]°. The error goes to -0.31° in one occasion.

σ_m	$\lvert e \rvert_x$	σ_x	$\lvert e \rvert_y$	σ_y	$\lvert e \rvert_\theta$	σ_{theta}	*It*	*Suc*
0.01	0.051	0.052	0.021	0.017	0.046	0.044	295.8	1.0
0.02	0.112	0.091	0.031	0.031	0.082	0.058	342.6	1.0
0.03	0.140	0.125	0.040	0.036	0.081	0.081	362.5	1.0
0.04	0.134	0.133	0.067	0.047	0.120	0.071	365.7	1.0
0.05	0.139	0.098	0.132	0.147	0.180	0.188	298.5	1.0
0.075	0.276	0.248	0.212	0.266	0.408	0.588	316.2	1.0
0.10	0.416	0.383	0.301	0.277	0.485	0.525	298.6	1.0
0.125	0.374	0.308	0.529	0.415	0.835	0.765	246.7	1.0
0.15	1.255	2.478	0.816	0.506	1.420	1.275	291.9	1.0
0.175	0.598	0.573	0.844	0.603	1.369	0.863	305.4	1.0
0.20	1.056	0.683	0.9611	0.705	2.186	1.641	294.4	0.96
0.225	2.242	3.426	1.5681	1.365	2.457	1.891	288.9	0.68
0.25	3.069	2.575	1.5849	1.252	1.826	1.384	264.3	0.60

Table 1. Accuracy of the algorithm for different levels of noise, true location (60,60,0).

7.1. Computational cost

When analyzing the computational cost of the algorithm, two different situations have to be considered:

• *Initial localization cycle.* In this situation, the algorithm explores the full state space until the stopping condition is reached. The time used to reach the stopping condition depends on the worst pose value considered as threshold in the stopping criteria. This time for the test example is around 4.5 seconds (in a T8300 duo core processor at 2.4 GHz with one core at execution). This time depends on the population set, the stopping criteria and the sensed area. The sensor perception estimation is done by ray tracing on the environment map and the estimation cost tends to grow with the size of the observed area since the algorithm concentrates its exploration in feasible areas. At the end of this first perception cycle, the feasible localization areas are determined.

•*Re-localization cycle*. Once the algorithm detects that the whole initial population has converged, the population set is decreased to 30 elements. For this population set the stopping condition is reached very fast and the computational cost decreases to 45 milliseconds per iteration.

These times allow the algorithm to be used on line except at the initial cycle.

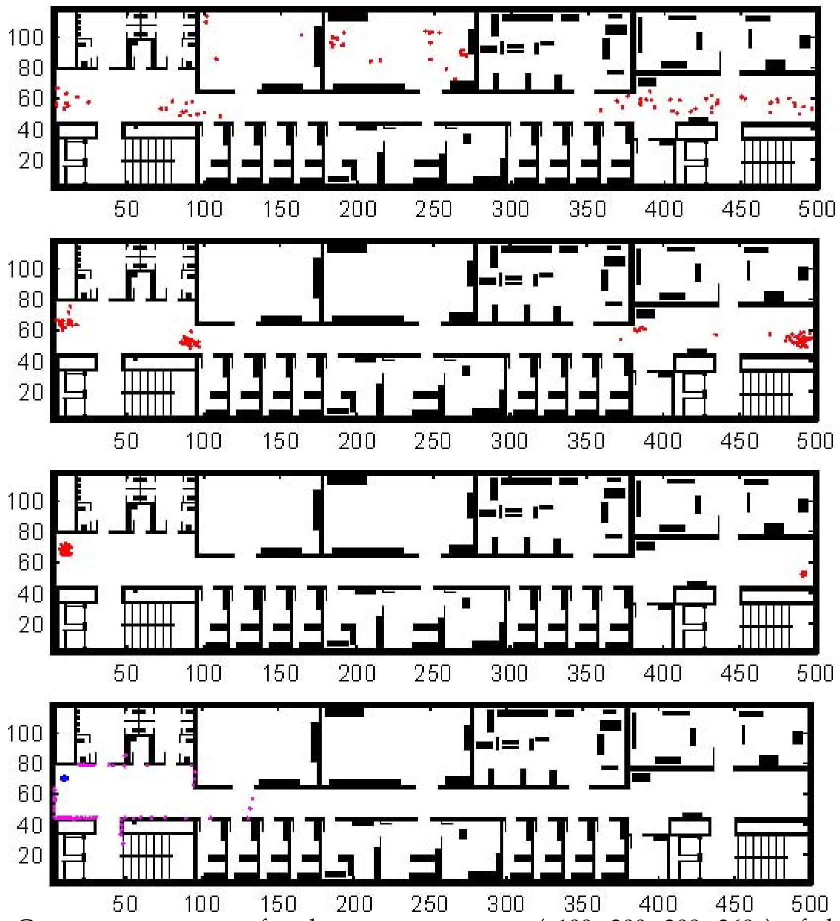


Fig. 10. Convergence process for the worst case pose (100, 200, 300, 360) of the first perception cycle.

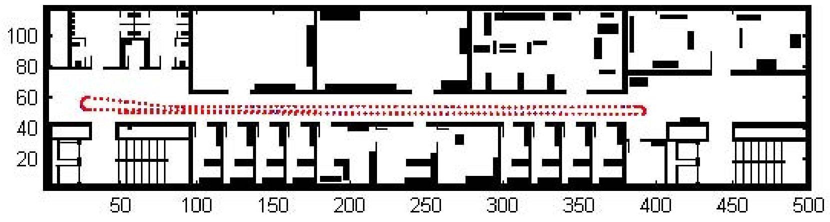


Fig. 11. Real and estimated trajectories along the central corridor.

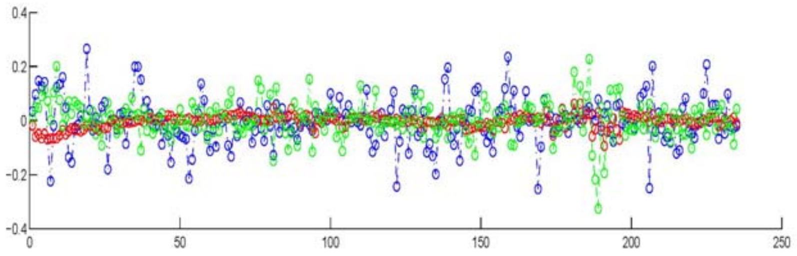


Fig. 12. Errors of the trajectory of Fig. 11 (in cells in x and y, and in degrees in θ).

8. Conclusion

One of the most interesting properties of the RDE (Rejection Differential Evolution) filter is its efficiency in terms of convergence speed. The algorithm is able to exploit the perceptual information to converge in the initial perception cycle if, from the initial pose, the robot perceives any distinctive characteristic of its environment. A second characteristic is the low number of pose solutions required to successfully localize the robot. For the test environment under consideration, a population of 100-150 elements is enough, except in non-informative situations where the number of hypotheses can grow up very fast and a certain number of pose solutions is required to maintain a feasible area.
The number of population elements required to avoid the premature elimination of feasible hypotheses has not been determined theoretically in the evolutive algorithms field, but in our experimental tests, a number between 10 and 25 poses is required to maintain all feasible hypothesis. In case of non informative situations where the sensors only let the robot perceive a small part of the environment (for instance, when a robot is in a corner) the potential number of hypotheses can rise very fast, which originates that the algorithm fails when using a normal pose set size. This problem can be addressed in two ways: turning the robot until a maximum environment area is perceived by the sensors or to detect the uninformative situation and increase the pose set size. The first approach is easier and requires less computational resources.
As in the majority of the population-based optimization methods, the algorithm robustness increases with the population set size. If we consider the effect of the population size on the accuracy of the algorithm, we need to consider the explored number of poses, since the total number of explored poses is roughly speaking the product of the iteration number and the population size. But in our test, the accuracy is maintained up to a certain number of explored poses. This behavior differs completely from Monte Carlo method. As noticed by several authors (Doucet, 1998; Liu & Chen, 1998), the basic Monte Carlo filter performs poorly if the proposal distribution, which is used to generate samples, places not enough samples in regions where the desired posterior probability distribution is large. This problem has practical importance because of time limitations existing in on-line applications.
The use of a rejection threshold and a stopping criteria adjusted to the statistical characteristics of the objective function allows us to decrease considerably the population size while maintaining the accuracy level of the method. In previous works, a minimum population set of 250 – 300 elements were required, while in the RDE version a population

of 100 has proved to be sufficient for the environment under consideration. At initial stages, the algorithm has to evaluate the whole environment map and the initial number of samples is related to the environment area and the area perceived with sensors. If the perceived area is big, the possible number of hypotheses required for the environment can be reduced and, consequently, the population required.
A significant characteristic of the method is the possibility of stopping the algorithm convergence to avoid the premature elimination of feasible hypotheses until posterior motion-perception cycles can add a significant statistical evidence to achieve the convergence to one pose. This characteristic is critical when the initial robot's pose is localized in a repetitive place, such as an office, without singular characteristics and the algorithm needs to detect the feasible hypotheses and to maintain them until distinctive characteristics are perceived in posterior perception cycles.
Due to the stochastic nature of the algorithm search of the best robot's pose estimate, the algorithm is able to cope with a high level of sensor noise with low degradation of the estimation results. The algorithm is easy to implement and the computational cost makes it able to operate on line even in relatively big areas.

9. References

Arras, K.; Castellanos, J. A. & Siegwart, R. (2002). Feature-based multi-hypothesis localization and tracking for mobile robots using geometric constraints, In: *Proc. of the Int. Conference on Robotics and Automation ICRA-02*, pp 1371-1377 , Washington D.C., USA

Austin, D. J.; & Jensfelt, P. (2000). Using multiple Gaussian hypotheses to represent probability distributions for mobile robot localization, In: *Proc. of the Int. Conference on Robotics and Automation ICRA-00*, pp 1036-1041, San Francisco, CA, USA

Branke, J.; Schmidt, C. & Schmeck, H. (2001). Efficient fitness estimation in noisy environments, In: *Proc. of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pp 243-250, Morgan Kaufmann

Burgard, W.; Fox, D.; Henning, D.; & Schmidt,T. (1996). Estimating the absolute position of a mobile robot using position probability drids, In: *Proc. of the National Conference on Artificial Intelligence AAAI-96*, pp 896-901, Portland, Oregon, USA

Cox, I. J. (1991). Blanche-An Experiment in guidance and navigation of an autonomous robot vehicle, In: *IEEE Transactions on Robotics and Automation*, **7**, pp 193-204

Cox, I. J.; & Leonard, J. J. (1994). Modeling a dynamic environment using a Bayesian multi hypothesis approach. *Artificial Intelligence*, 66, pp 311-44

Crowley, J. L. (1989). World modelling and position estimation for a mobile robot using ultrasonic ranging, In: *IEEE International Conference on Robotics and Automation*, Scottsdale, AZ

Dellaert, F.; Fox, D.; Burgard, W.; & Thrun, S. (1999). Monte Carlo Localization for Mobile Robots, In: *Proceedings of the 1999 International Conference on Robotics and Automation*, pp. 1322-1328

Doucet, A. (1998). On sequential simulation-based methods for Bayesian filtering. *Technical Report CUED/FINFENG/TR 31'*, Cambridge University, Dept. of Engineering, Cambridge, UK

Fox, D., Burgard, W.; & Thrun, S. (1999). Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, **11**, pp 391-427

Garrido, S.; Moreno, L. & Salichs, M.A. (1998). Non linear on line identification of dynamic systems with restricted genetic optimization, In: *Proceedings of the 6th European Congress on Intelligent Techniques and Soft Computing, EUFIT*, pp. 423-428

Garrido, S. & Moreno, L. (2001). Learning adaptive Parameters with Restricted Genetic Optimization, In: *Bio-inspired Applications of Connectionism: 6th International Work-Conference on Artifical and Natural Neural Networks, IWANN2001*, pp 612-620, Springer-Verlag

Goldberg, D. E. (1989). *Genetic algorithm in Search, Optimization, and Machine Learning*. Addison Wesley Publishing Company

He, J. & Kang, L. (1999). On the convergence rates of genetic algorithms. *Theoretical Computer Science*, 229, pp 23-39

Jensfelt, P. & Kristensen, S. (1999). Active global localisation for a mobile robot using multiple hypothesis tracking, In: *Proc. IJCAI Workshop on Reasoning with Uncertainty in Robot Navigation*, pp 13-22, Stockholm, Sweden.

Jensfelt, P.; Wijk, O.; Austin, D. J. & Andersson, M. (2000). Experiments on augmenting condensation for mobile robot localization, In: *Proc. of the Int. Conference on Robotics and Automation ICRA-00*, pp 2518-2524, San Francisco, CA, USA

Jensfelt, P. (2001). *Approaches to mobile robot localization in indoor environments*. Doctoral Thesis, Royal Institute of Technology, Sweden

Kalos, M. H. & Whitlock, P.A. (1986). *Monte Carlo Methods. Volume I: Basics*, John Wiley and Sons

Krink, T.; Filipic, B.; Fogel, G. & Thomsen, R. (2004). Noisy Optimization Problems - A Particular Challenge for Differential Evolution, In: *Proc. of the 2004 IEEE Congress on Evolutionary Computation*, pp 332-339, IEEE Press

Liu, J. & Chen, R. (1998). Sequential Monte Carlo methods for dynamic systems. *Journal Amer. Statis. Assoc.*, 93, pp 1032-1044

Leonard, J.J. & Durrant-White H.F. (1992). *Directed sonar sensing for mobile robot navigation*. Kluwer Academic Publishers

Lenser, S. & Veloso, M. (2000). Sensor resetting localization for poorly modelled mobile robots, In: *Proc. IEEE International Conference on Robotics and Automation (ICRA-2000)*, San Francisco, C.A.

Markon, S., Arnold, D. V.; Baeck, T.; Bielstein, T. & Beyer, H.- G. (2001). Thresholding- a selection operator for noisy ES., In: *Proc. of the 2001 IEEE Congress on Evolutionary Computation*, pp 465-472, IEEE Press

Moreno, L.; Garrido, S. & Muñoz, M. L. (2006). Evolutionary filter for robust mobile robot localization, In: *Robotics and Autonomous Systems*, 54, pp 590-600

Reuter, J. (2000). Mobile robot self-localization using PDAB, In: *Proc. IEEE International Conference on Robotics and Automation (ICRA-2000)*, San Francisco, C.A.

Roumeliotis, S. I. & Bekey, G.A. (2000). Bayesian estimation and Kalman filtering: A unified framework for mobile robot localization, In: *Proc. IEEE International Conference on Robotics and Automation (ICRA-2000)* , pp 2985-2992, San Francisco

Storn, R. & Price, K. (1995). Differential Evolution- A simple and efficient adaptive scheme for global optimization over continuous spaces. *Technical Report TR-95-012*, March

Thrun, S. (1998). Bayesian landmark learning for mobile robot localization, *Machine Learning*, 33(1)

Thrun, S; ,Fox, D.; Burgard, W. & Dellaert, F. (2001). Robust Monte Carlo localization for mobile robots. *Artificial Intelligence*, 128, pp 99-141

Vahdat, A. R.; Ashrafoddin, N. N. & Ghidary, S.S. (2007). Mobile Robot Global Localization using Differential Evolution and Particle Swarm Optimization, In: *Proc. IEEE Congress on Evolutionary Computation (CEC-2007)*, pp 1527-1534

Reliable Localization Systems including GNSS Bias Correction

Pierre Delmas, Christophe Debain
Cemagref
France

Roland Chapuis
LASMEA
France

Cédric Tessier
Effidence
France

1. Introduction

GNSS (Global Navigation Satellite System) system is the most famous system to realize localization of mobile robots. It is able to provide a global position everywhere in the world.
Moreover, as it provides a global position, it is not necessary to place the robot in known or structured environments (i.e. where several landmarks are available) to localize it unlike local positioning systems. So, the GNSS system has became an unavoidable system to guide automatically mobile robots.
Generally, GPS-RTK (Real Time Kinematic) sensors are used to localize and guide automatically mobile robots in outdoor environment. This type of sensor has a precision about few centimeters. However, they are expensive and difficult to use because they need differential correction to improve their precision. Consequently, an other GPS receiver is necessary as a reference. Moreover, their accurate measure is not always available because of GPS signal losses or multipaths. In this case, their accuracy to within 2 centimeters which is their main advantage, is not always available. So, it is impossible to use GPS-RTK sensors alone to have an effective localization system in the context of autonomous guidance. It must be used with other sensors to insure the localization when GPS signal is not available (Pein et al., 2006).
Some research makes an effective localization system for mobile robots using GPS-RTK with other satellite sensors. However, the dissemination of automatic guidance system in outdoor environment can use a low-cost sensors as natural GPS or GPS with a differential correction (WAAS or EGNOS). This type of sensors are less expensive than GPS-RTK and easy to use, because the operator has just to use and manage a single receiver. This last one has an accuracy between 1 and 3 meters with WAAS or EGNOS differential correction. To enhance their precision, natural GPS data can be fused with other exteroceptive data as it is done in (Tessier et al., 2006a). Generally, these localization systems are based on a Kalman Filter (KF) because

autonomous guidance systems use a mono hypothesis localization estimation.
Nevertheless, there is an important assumption to use a KF, the errors of each sensor must be zero-mean, Gaussian and white process. If this assumption is not true, the estimation of position will be biased. Indeed, for every new observations of sensors data, the precision of localization (i.e. size of confidence ellipsis built with the covariance matrix) increases but the estimated position diverges from the true position (figure 1).

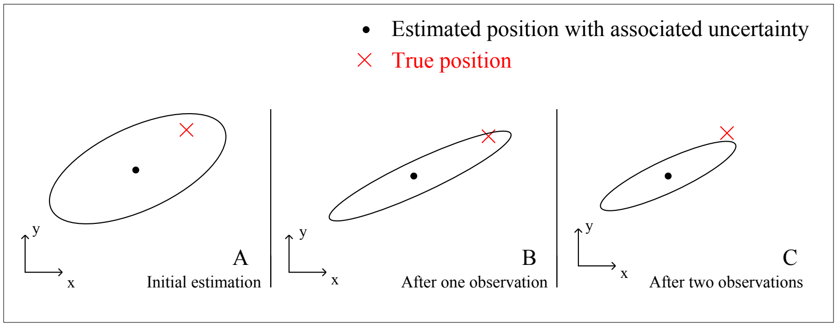


Fig. 1. Localization results after the fusion of two observations coming from a colored process (A at t_0; B at t_1; C at t_2 with $t_0 < t_1 < t_2$)

In this figure, the situation C shows the true position is out of estimated position (position with its associated uncertainty). So, it becomes impossible to check the reliability of the localization. Consequently, the estimated position becomes useless, the mobile robot doesn't know where it is truly and it is impossible to accurately guide the vehicle. This is critical and dangerous situation which must be imperatively avoided. This critical situation appears for the localization systems which use GNSS receiver because generally they don't take care of the GNSS error characteristics. So, we propose in this paper a method to improve GNSS receiver precision and accuracy in the context of autonomous guidance.

In section II, we suggest an analysis of GNSS bias characterisation, one process to establish a model of it, and a solution to detect disturbances in GNSS data. Then, in the section III, we show how to improve easily a KF by inserting the prediction model and the condition. We will see the result of this method showing the robustness of GNSS bias modeling and the improvement of our localization system in the last section.

2. Characterisation of the GNSS error

The observations of GNSS data cause an unreliable estimated position. To better understand the problem, we propose an analysis of GNSS data.

2.1 Data Analysis from a GNSS receiver

The GNSS systems are based on localization by triangulation. The satellites send a message with the information (time and satellite ephemeris) allowing the receivers on the Earth calculate their position. Unfortunately, the GNSS system is not perfect. Many measurement errors cause a bad localization like satellite clock errors, ephemeris errors, atmospheric (ionosphere and troposphere) errors, multipath effects, receiver noise and resolution (for more information see (Kaplan, 1996)). Table 1 summarizes these errors with their characteristics.

Error sources	Precision	Errors Characteristic
Satellites clock errors	± 2 meters	stable
Satellites ephemeris errors	± 2.5 meters	stable
Ionospheric effects	± 5 meters	stable
Tropospheric effects	± 0.5 meters	stable
Multipath effects	± 1 meter	unpredictable
Receiver noise and resolution	± 1 meter	unpredictable

Table 1. GNSS Error Sources.

We see that most of error sources are stable process (i.e. they evolve slowly). If we compare the low-cost GPS data with the true position coming from a GPS-RTK receiver (figure 2), we see that the natural GPS error seems to be constant. If we observe the auto-correlation of GPS longitude error between each iteration in static condition (figure 3), we realize this error is very correlated between successive iterations (it is the same case for GPS latitude error). It means the GPS error is a non white process. It is commonly named the GPS bias.
If we estimate the position with only the GNSS observations (position and associated covariance given by GNSS receiver) thanks to a KF, we obtain very quickly an unreliable estimated position (figure 4). It is always possible to increase the covariance of GNSS data. However, although this solution increases the reliability of GNSS measurement, the real problem is the GNSS error is a stochastical process. As soon as the position is calculated by KF thanks to GNSS observations, the estimated uncertainty decreases too much and consequently, the system becomes unreliable. So, it is necessary to know the value of GNSS error every time so as to have a reliable estimated position.

2.2 GNSS error modelizing

We have seen the main problem of inefficiency of localization system based on a KF is the non white GNSS error. So, we must find a model which describes this bias.
Some researcher proposed to find global criteria determining GNSS error (Nienaber & Diekhans, 2006). These criteria consider only mean and standard deviation of the error in static and dynamic condition during 24 hours. However, many applications (agricultural tasks, automatic guidance, ...) run usually several hours and the characteristics (means and variance) of GNSS error are not the same for 24 hours. Table 2 represents the mean and standard deviation of data for three different moments. So, it is impossible to determine a reliable model with only mean and variance of GNSS error.

Database	Longitude (m)		Latitude (m)	
	mean	RMS	mean	RMS
Database 1 (2 hours)	1.05	1.27	0.24	1.81
Database 2 (2 hours)	2.72	1.31	-1.62	1.24
Database 3 (2 hours)	1.46	2.03	-0.70	1.26

Table 2. Means and Variances of data at three different moments.

Another solution is to estimate the GNSS error by inserting it in the state vector of KF as it is described in (Perera et al., 2006). A method to determine bias sensor online is proposed. The

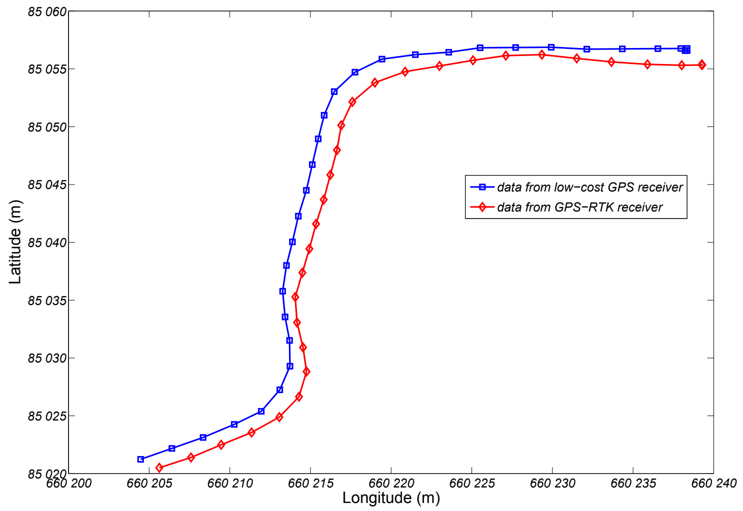


Fig. 2. Example of GPS data coming from low-cost GPS (blue) and RTK-GPS (red)

error is determined thanks to an exteroceptive sensor. We propose to improve this solution inserting a more accurate prediction model for GNSS error.
We make the assumption that the GNSS data is composed by position (x,y), an stochastical process ((b_x,b_y) the associated GNSS bias) and a zero-mean, white, Gaussian noise (ϵ_x,ϵ_y) like (1). So, the aim is to look for a model which is able to predict the GNSS bias. This model must be determined to have a residual error which is zero-mean, white, Gaussian process. Then, this bias prediction model will be inserted in the KF to determine it and the position in the same time (see section III). The observation error of KF becomes zero-mean, white, Gaussian and doesn't drift the localization.

$$Data_{gnss} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} b_x \\ b_y \end{pmatrix} + \begin{pmatrix} \epsilon_x \\ \epsilon_y \end{pmatrix} \tag{1}$$

We have seen GNSS bias is a stochastical model. To answer the problematic, we choose to use an AutoRegressive process described in (Laneurit et al., 2006) by Laneurit. Indeed, the AR process is a filter which has a zero-mean, white, Gaussian process in input and the stochastical process to determine in output. It is often used for a vocal identification like in (Kotnik & Kačič, 2007). It is formulated in Z-transform by (2).

$$F_{AR} = \frac{Y(z)}{\epsilon(z)} = \frac{1}{1 + \alpha_1 z^{-1} + \alpha_2 z^{-2} + \ldots + \alpha_p z^{-p}} \tag{2}$$

with Y the stochastical signal, ϵ the zero-mean, white, Gaussian process, α and p respectively the parameters and the order of AR process. In our case, the expression (2) becomes (3) in discrete domain.

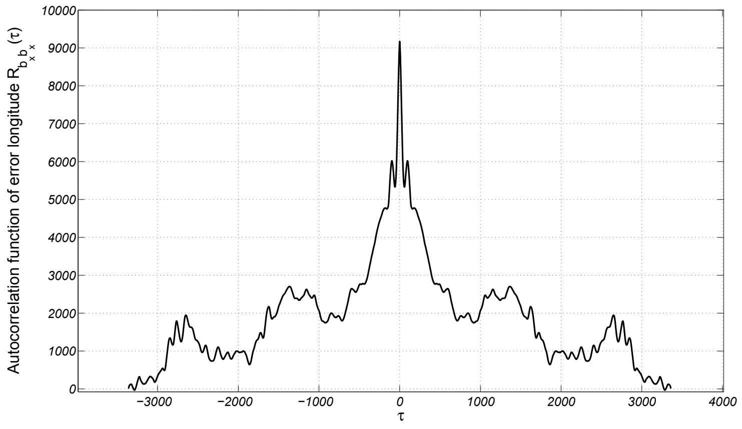


Fig. 3. Autocorrelation function of longitude error

$$b_{k+1} = \sum_{i=1}^{p} -\alpha_i.b_{k-i+1} + \epsilon_k, k \in [1 : N-1], p \in \aleph^* \tag{3}$$

with b_k the GNSS bias at k^{th} iteration. Now, it must find the parameters and the order of the AR process. This determination is made thanks to preliminary database of GNSS receiver. To always have a stationary model, the AR parameters are calculated by Burg method (Burg, 1975). Now, for the choice of AR process order, we take the order at the point where the power of ϵ stops decreasing significantly. The figure 5 represents the power of residual error between the real and estimated GNSS bias value for different AR order. Indeed, if AR process order is too reduced, the process won't represent the intrinsic properties due to GNSS signal. However, if AR process order is too big, the process will represent the properties due to signal noise. Other criteria exist as AIC (Akaike Information Criteria) and BIC (Bayesian Information Criteria) (Akaike, 1973) and (Schwarz, n.d.).
We have established the prediction model of GNSS bias thanks to AR process. But, the losses of one satellite may cause a disruption of GNSS observation. The figure 6 shows the GNSS observations change abruptly (about 50 centimeters) at time t=6160s. Another important example of data disturbances is the multipath effects. When the receiver is close to an obstacle, the GNSS signal may reflect on this obstacle before to be received. This quick evolution causes disturbances on the prediction model. Consequently, to always have a GNSS localization reliable, these phenomena must be detected so as to reset the GNSS bias estimation.
The proposed idea to solve this problem is to compare the predicted GNSS data with the observation thanks to the Mahalanobis distance in (Laneurit, 2006). Generally, this distance is used to detect spurious data like in (Wu & Zhang, 2006). Contrary to Euclidean distance, it takes care of data correlation. It is formalized by (4).

$$d = \sqrt{(Z_{gnss,k} - \hat{Z}_k)^T.(\hat{R}_k + R_{gnss})^{-1}.(Z_{gnss,k} - \hat{Z}_k)} \tag{4}$$

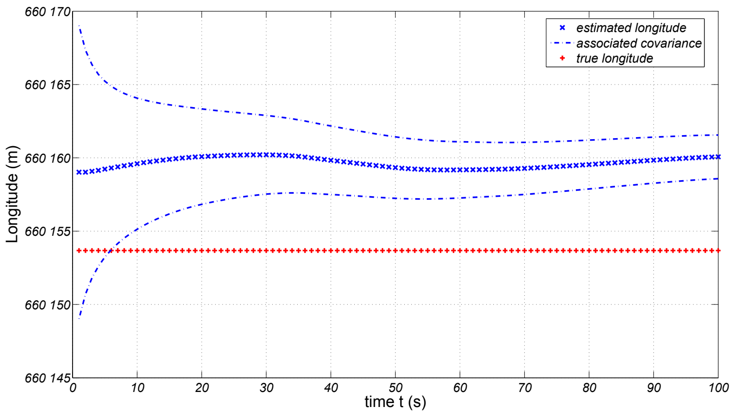


Fig. 4. Longitude estimation by KF with only GNSS observations with variance given by GNSS receiver (at t=5, the estimated position becomes false)

with $Z_{gnss,k}$ the GNSS observation in time k, the matrix R_{gnss} is the covariance matrix of the GNSS observation, $\hat{Z}$ and $\hat{R}$ are respectively the predicted value of GNSS data and its associated covariance. If the Mahalanobis distance between predicted and observed GNSS data is bigger than three, the GNSS localization system must be reset. Now, we are able to predict the stochastical GNSS error so as to determine it with the position thanks to KF. That leads to have only an observation error which can be considered like zero-mean, white, Gaussian noise. We have all information to create an localization system reliable using GNSS system.

3. Integration of GNSS error model in KF

In the previous part, we have established the prediction model of GNSS bias for latitude and longitude. Now, we will see how it is inserted in the KF. For GNSS localization, the **state vector** of KF is X defined by (5).

$$X_k = \left(x_k, y_k, b_{x,k}, \ldots, b_{x,k-p_x-1}, b_{y,k}, \ldots, b_{y,k-p_y-1}\right)^T \tag{5}$$

with x_k, y_k the estimation of Cartesian coordinates, $b_{x,k}$, $b_{y,k}$ bias of respectively longitude and latitude at k^{th} iteration and p_x and p_y the prediction model order of respectively for the longitude and the latitude model. The choice of AR process order for the prediction bias model is very important because it determines the size of state vector of KF (in this case, $Size(X) = 2 + p_x + p_y$). If the size of state vector is too big, the computing time will be too long for real-time applications and the automatic guidance can lost its stability.

The Kalman filter is composed in three parts : initialization, Prediction level and Correction Level. We will describe each part for the bias estimation and add a new part for the detection of disruption GNSS data.

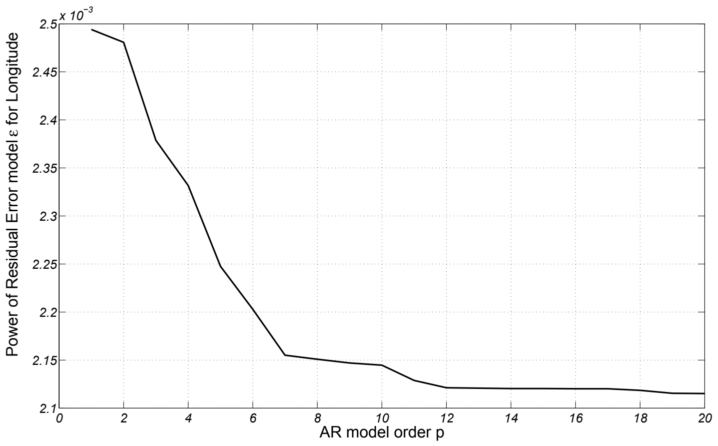


Fig. 5. Power of error predicted model $\epsilon(p)$ for longitude. The best order is 8.

Initialization : Generally, the GNSS bias and the position are unknown. In that case, it is impossible to determine them. Indeed, the GNSS observation represents the sum of them and it is impossible with only GNSS data to separate them. We have an observation problem. We are obliged to know one of them to begin the estimation. To show the precision and the reliability of our method, we assume to begin the estimation with the known position (5cm precision) and the bias with a precision of 10m. In our localization systems, we use data coming from exteroceptive sensors. Then, the system can localize it without this information until the new reset.

Prediction Level : In the previous section, we have seen equation of bias evolution (3). This equation is the prediction equation of GNSS bias for the KF. So, the state evolution matrix A is composed of three submatrices : $A_{x,y}$ for Cartesian position; A_{b_x} and A_{b_x} for respectively the longitude and the latitude GNSS bias. The AR parameters are easily inserted in the submatrix A_{b_x} for example like (6).

$$A = \begin{pmatrix} A_{x,y} & 0 & 0 \\ 0 & A_{b_x} & 0 \\ 0 & 0 & A_{b_y} \end{pmatrix} \; with \; A_{b_x} = \begin{pmatrix} -\alpha_1 & \cdots & \cdots & -\alpha_{p_x} \\ 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 \end{pmatrix} \tag{6}$$

The variance of prediction equation ϵ is inserted in the prediction covariance matrix Q.

Detection of GNSS data disturbances : In the previous part, we have established a condition to detect disturbances of GNSS data. This condition is based on the Mahalanobis distance. This detection must be before the correction level to not degrade the estimation and after the prediction level of KF to allow the comparison between predicted and observed GNSS data. If the condition is respected, the estimation continues else the estimation must be initialized.

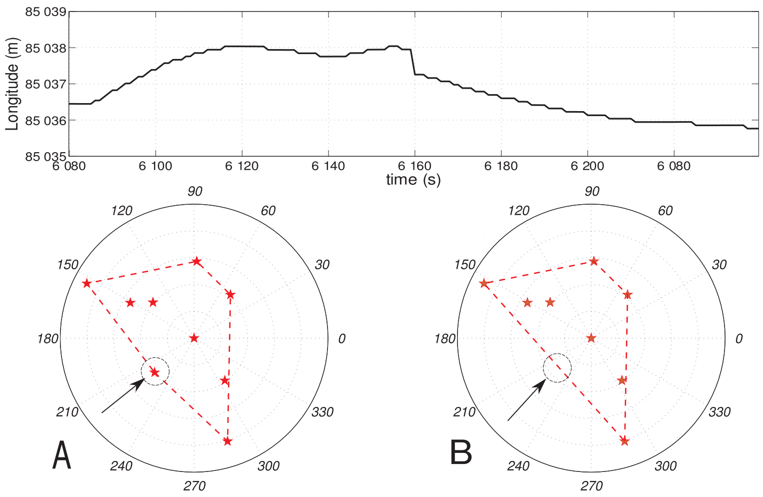


Fig. 6. Constellation influence on GNSS data. The situation A is the satellite constellation at t=6159s and the situation B at t=6160s. Between the two situations, the pointed satellite disappears.

Correction Level : Now, the observed data is not only composed by the position but it is the sum of the position and the associated bias. We have seen in the initialization, that causes problem of observability. But, we make assumption the position is known with other sensor in the initialization. The state observation matrix for the GNSS H_{gnss} is (7).

$$Z_{gnss,k} = H_{gnss} * X \; with \; H_{gnss} = \begin{pmatrix} 1\,0\,1\,0\,0 \dots 0 \\ 0\,1\,0\,1\,0 \dots 0 \end{pmatrix} \tag{7}$$

To determine the power of their noise, we have made assumption that position data and bias are signal which are only composed of low-frequency component so we have filtered data signal by a high-pass filter and we have estimated the power of residual signal is equal to those of observation error. This power is integrated in observation covariance matrix R_{gnss}. We see we have an observability problem. To solve this problem, we must know at a moment the position or the bias. In the initialization of localization systems, we know the true position thanks to GPS-RTK. It is possible to determine position by exteroceptive sensors by example. To resume, we obtain the figure 7.

4. Results

In this part, we present results of localization using GNSS with bias correction as we propose in section III. First, we will see the localization results with only GNSS receiver. Then, we will present our localization system for autonomous guidance and the improvement of this method.

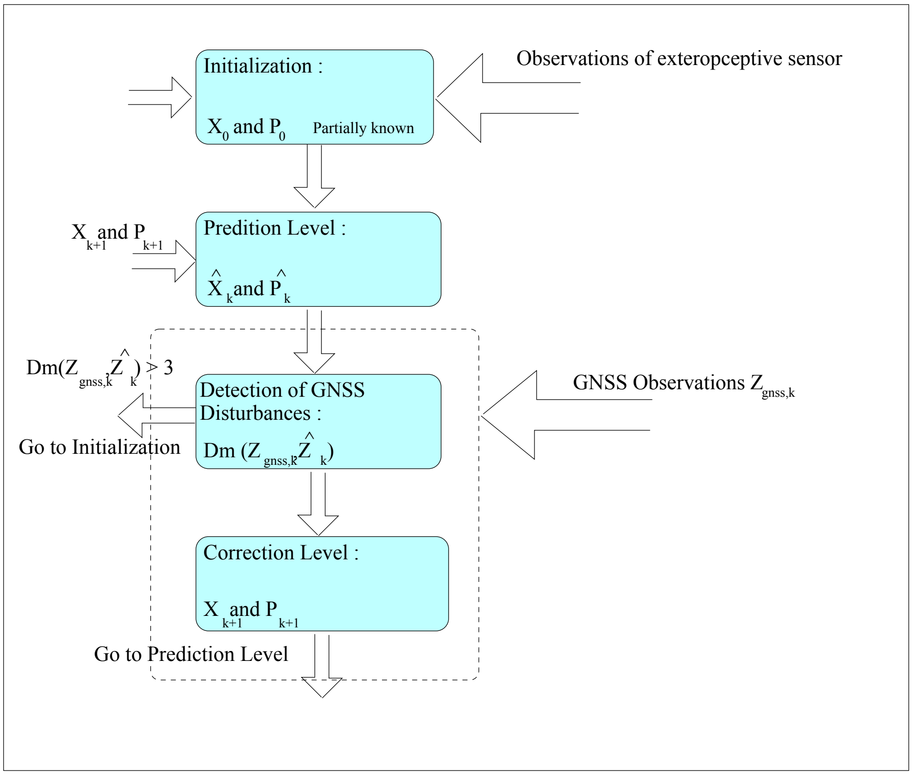

Initialization :
X_0 and P_0 Partially known
Observations of exteropceptive sensor
X_{k+1} and P_{k+1}
Predition Level :
$\hat{X}_k$ and $\hat{P}_k$
$Dm(Z_{gnss,k}, \hat{Z}_k) > 3$
Go to Initialization
Detection of GNSS Disturbances :
$Dm(Z_{gnss,k}, \hat{Z}_k)$
GNSS Observations $Z_{gnss,k}$
Correction Level :
X_{k+1} and P_{k+1}
Go to Prediction Level

Fig. 7. Principle of Kalman filter with the detection of GNSS disturbances

Localization with GNSS alone : This test is done with a low-cost GPS receiver Holux GR213. We have established the prediction model in off-line before the test. Then, we have made the test in static condition during two hours for different orders of AR process without the condition to detect GNSS disturbances. The purpose of this test is to show the improvement of precision and the reliability of the method for a long period. Table 3 summarizes the results. In this table, we see that the choice of AR order is not an important parameter as it is shown by the little difference between each model. We can use the AR1 process which is the easiest model and the best for computing time. The figure 8 represents the estimated longitude with our method. We see the estimation is reliable for a long time in spite of GNSS disturbances. This result shows the robustness of prediction model. So for the second test, based on the evaluation of the quality of a localization process using a low cost GNSS receiver, we will use the AR1 process to establish a model of the GNSS stochastical error.

Localization for autonomous guidance : We have inserted the bias modeling in our localization system used to automatically guide our small mobile robot AROCO (see figure11). The sensors on-board the vehicle, amongst others, include a fiber-optic gyroscope (FOG) [KVH DSP 3000], rear and front wheel encoders, a low-cost GPS (5*m* accuracy), a SICK PLS200 laser measurement system and a CCD camera (SONY VL500). The SICK PLS200 provides range

Bias model	Error for longitude (m)			Error for latitude (m)		
	mean	max	RMS	mean	max	RMS
None (variance of GNSS error given by the receiver)	1.31	3.33	0.53	-0.91	1.42	0.34
AR 1 Process	-0.015	0.040	0.009	-0.010	0.045	0.011
AR 2 Process	-0.009	0.249	0.066	-0.009	0.29	0.078
AR 3 Process	-0.022	0.197	0.081	-0.147	0.356	0.094
AR 4 Process	0.007	0.217	0.098	-0.194	0.424	0.122
AR 5 Process	0.069	0.310	0.118	-0.233	0.488	0.143

Table 3. Error for Estimated position in static condition.

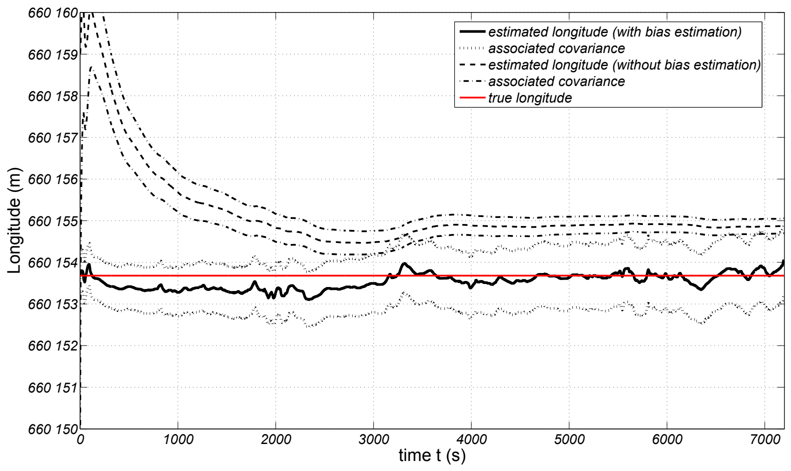


Fig. 8. Longitude estimation by KF taking care of GNSS stochastical error

measurements to object ahead at 0.5^o intervals over a span of 180^o in one scan. The scanned data arrive every $80ms$. The SONY VL500 gives 7.5 640x480 YUV422 images per second. Finally, the robot is equipped with two on-board computer systems. The first running Linux RTAI is a low level system responsible for driving engine. The second running Linux is a high level system where our localization algorithm runs and sends control commands to the first PC.

Our localization system (developed by Tessier (Tessier et al., 2006a)) fuses local information with cheap GPS data (Tessier et al., 2006b). It estimates the vehicle's state (absolute position and orientation with their incertitude and their reliability) in a world reference frame represented by dynamic GIS (Geographic Information System). Local information given by landmarks detection allows the system to improve the position of localization given by natural GPS. Thanks to the use of proprioceptive and local information, an automatic guidance system can estimate the reliability of its localization (Tessier, Berducat, Chapuis & Bonnet, 2007). Before the correction of the GNSS bias, the only way to use natural GPS in our guidance sys-

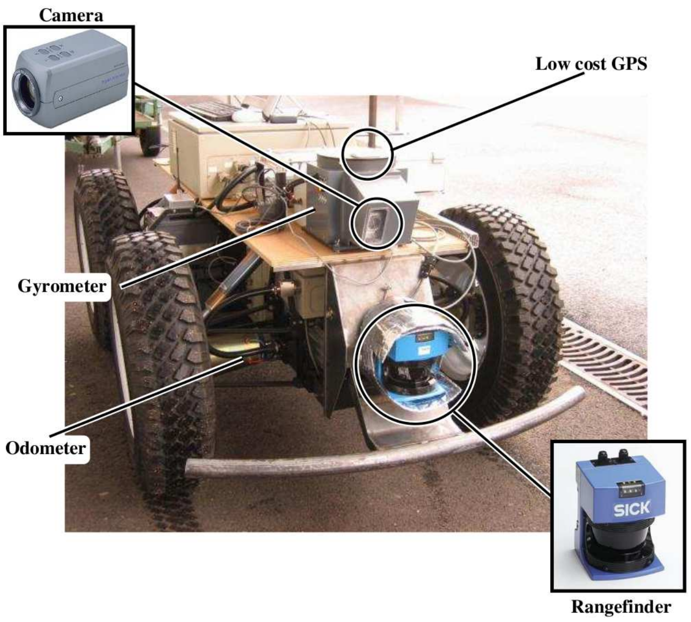


Fig. 9. Mobile robot used in experiments.

tem with an admissible level of reliability was to increase the GPS given by receiver. With our correction of GPS bias, the precision and the reliability of natural GPS are enough high to increase the performance of the guidance process.
Numerous real experimentations were made in outdoor environment to measure system capabilities and to attest the efficiency of our approach. We test it on our vehicle along the trajectory presented in figure (figure 11). Due to the presence of trees along the trajectory, we have satellite losses and lots of changes of satellite constellation.Thus, the vehicle moved automatically in wooded environments, in clear area, close to buildings, in hilly ground and under difficult climatic conditions (i.e. sunny weather, hailstorm, . . .). The length of trajectory is about 400 meters and the speed of the vehicle is maintained at approximatively 3m/s. The purpose of GPS bias estimation is to improve precision and particularly the reliability of our localization system. In our experimentation, we observe the estimated position of vehicle is more reliable than the past experimentation (the detection of landmarks is always successful contrary to the past experimentation). However, the localization system must be sometimes initialised because of GNSS disturbances and must use exteroceptive sensor (every 1 minute approximatively). If the landmarks is not available for a long time, the localization system doesn't certify its reliability, however the drift of estimated position due to GNSS disturbances is very small in comparison at the past.
• For this experiment, the vehicle succeeds in tracking the pre-defined trajectory with a good accuracy. However to check the repetitiveness and the sturdiness of the proposed approach, the vehicle reproduces faithfully the trajectory during 10 laps without stopping. Seven checkpoints have been placed on the path to measure the positioning errors. Those errors are presented in Table 4. They corresponds to the real lateral deviation of the vehicle with the reference path during the autonomous driving (i.e. localization and guidance). As we can notice,

	A	B	C	D	E	F	G
mean (cm)	-1.7	-0.2	0.5	-0.2	0	-0.7	-0.1
std (cm)	4.7	3.2	3.7	3.1	3.9	1.4	2.2
max (cm)	9.2	6	7.3	4.5	6	3.6	5.2

Table 4. Real lateral deviation during path following for 10 laps.

the system is very accurate, the max error is below $10cm$. The use of a multi-sensor system with an active search approach permits to locate the vehicle accurately with real-time constraints.
• Even in presence of disturbances, the system behaves correctly. The day of this experimentation, a strong downpour of hail fell. This disturbed the landmarks detection. Indeed, some hailstones are detected by the range-finder (figure 10). Thanks to the focusing process, this problem is attenuated and permits to identify some landmarks. Nevertheless, the system failed in detecting other landmarks because they are masked by the hailstones. Consequently, the reliability decreases and the vehicle speed slackens off. However, the system searches more easily recognizable landmarks (lane side with the camera, wall with the range-finder) to strengthen the estimation (i.e. increase the reliability) and boost the speed. At the end, the *SLAG approach is robust*.

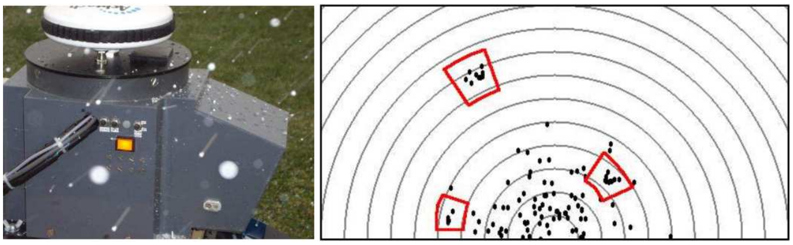
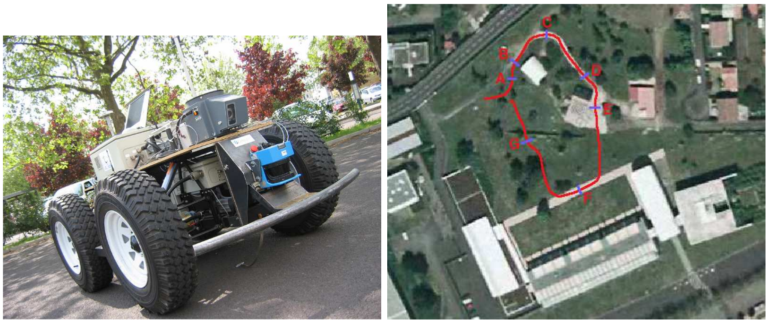


Fig. 10. (left) A strong hailstorm during our experimentation.
(right) Range-finder measurement with three trees and their region of interest.



(a) The vehicle Aroco (b) Trajectory for the test

Fig. 11. Trajectory for the test with vehicle Aroco

5. Conclusion

In this chapter, we have presented a method to improve localization systems based to data association with GNSS receiver. This method increases the **precision** and the **reliability** of localization based on an Kalman filter. It consists to take care the characteristics of GNSS error. This error is an unpredictive stochastic process and it drifts the estimated position which is calculated by a Kalman Filter. The developed idea is to establish a prediction model of GNSS bias and to insert it in the localization system so as to modify the observation error from low-cost GNSS receiver to zero-mean, white, Gaussian noise. We have seen a possible model of GNSS error is Autoregressive process. We have determined its parameters and its order. Then, we have shown how this model is inserted in the Kalman Filter. However, the bias estimation needs to have sometimes absolute data (position of landmark of the environment) coming from exteroceptive sensors. To do that we propose to use a multi-sensor system (Tessier, Debain, Chapuis & Chausse, 2007) in which landmarks detection is given by autonomous entities called "perceptive agents". The main weakness of this multi-agent fusion system is about the focusing process and the measure of the accuracy of the estimated vehicle's pose. Thanks to numerous experiments we noticed a strong robustness and a good accuracy of the guidance process allowing using it at high speed even in an environment with lots of elements like trees or buildings.

6. References

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle, *Proceeding of the* 2^{nd} *International Symposium on Information Theory*, Budapest, pp. 267–281.

Burg, J. (1975). Maximun entropy spectral analysis, *Departement de Geophysiques, Universite de Stanford, Stanford, Californie, USA* .

Kaplan, E. (1996). *Understanding GPS: Principles and Applications*, Artech House.

Kotnik, B. & Kačič, Z. (2007). A noise robust feature extraction algorithm using joint wavelet packet subband decomposition and ar modeling of speech signals, *Signal Process* **Vol.87**: 1202–1223.

Laneurit, J. (2006). Perception multisensorielle pour la localisation de robot mobile en environnement extérieur, *Phd thesis, Universite Blaise Pascal - Clermont-Ferrand, France* .

Laneurit, J., Chapuis, R. & Chausse, F. (2006). Accurate vehicle positioning on a numerical map, *The International Journal of Control Automation and Systems* **Vol.3**(No.1): 15–31.

Nienaber, G. & Diekhans, N. (2006). Accuracy analysis of gps-based autoguidance systems, *VDI BERICHTE* **Vol. 1958**: 329.

Pein, C., Rőhrich, D., Skjodt, P. & McClure, J. (2006). Combining gps and camera steering (lps) systems: in field applications and experiences, *Proceeding of AgEng Bonn 2006, 64th VDI-MEG International Conference Agricultural Engineering*, Bonn, p. 1958.

Perera, L. D. L., Wijesoma, W. S. & Adams, M. D. (2006). The estimation theoretic sensor bias correction problem in map aided localization, *The International Journal of Robotics Research* **Vol.25**(No.7): 645–667.

Schwarz, G. (n.d.). Estimating the dimension of a model, *The Annals of Statistics* **Vol.6**(No.2): 461–464.

Tessier, C., Berducat, M., Chapuis, R. & Bonnet, S. (2007). Characterization of feature detection algorithms for a reliable vehicle localization, *Proceeding of 6th IFAC Symposium on Intelligent Autonomous Vehicles*, Toulouse, France.

Tessier, C., Debain, C., Chapuis, R. & Chausse, F. (2006a). Active perception strategy for vehicle localization and guidance, *Proceeding of the* 2*th* *IEEE International IEEE Conference on Cybernetics & Intelligent Systems and Robotics, Automation & Mechatronics*, Bangkok, Thailand, pp. 244–249.

Tessier, C., Debain, C., Chapuis, R. & Chausse, F. (2006b). Fusion of active detections for outdoors vehicle guidance, *Proceeding of the* 9*th* *International IEEE Conference on Information Fusion*, Florence, Italie, pp. 1914–1921.

Tessier, C., Debain, C., Chapuis, R. & Chausse, F. (2007). A cognitive perception system for autonomous vehicles, *Proceeding of COGIS'07 COGnitive systems with Interactive Sensors*, Standford University California, USA.

Wu, N. & Zhang, J. (2006). Factor-analysis based anomaly detection and clustering, *Decision Support Systems* **Vol.42**: 375–389.

Evaluation of aligning methods for landmark-based maps in visual SLAM

Mónica Ballesta, Óscar Reinoso, Arturo Gil,
Luis Payá and Miguel Juliá
Miguel Hernandez University of Elche
Spain

1. Introduction

Building maps is one of the most fundamental tasks for an autonomous robot. This robot should be able to construct a map of the environment and, at the same time, localize itself in it. This problem, known as Simultaneous Localization and Mapping (SLAM), has received great interest in the last years (Leonard & Durrant-Whyte, 1991).
In our particular case, the robots build their maps using the FastSLAM algorithm (Montemerlo et al., 2002). The main idea of the FastSLAM algorithm is to separate the two fundamental aspects of the SLAM problem: the estimate of the robot's pose and the estimate of the map. This algorithm uses a particle set that represents the uncertainty of the robot's pose (localization problem) meanwhile each particle has its own associated map (several individual estimates of the landmarks conditioned to the robot's path). The solution to the SLAM problem is performed by means of a sampling and particle generation process, in which the particles whose current observations do not fit with their associated map are eliminated. The FastSLAM algorithm has proved to be robust to false data association and it is able to represent models of non-linear movements in a reliable way (Howard, 2006).
In relation to the sensors used to build the maps, many authors use range sensors such as sonar (Kwak et al., 2008; Wijk & Christensen, 2000) or laser (Thrun, 2004; Triebel & Burgard, 2005). Nevertheless, there is an increasing interest on using cameras as sensors. This approach is denoted as visual SLAM (Valls Miro et al., 2006). These devices obtain a higher amount of information from the environment and they are less expensive than laser as well. Furthermore, 3D information can be obtained when using stereo cameras. These are the sensors used in this work.
Most visual SLAM approaches are landmark-based. These landmarks consist of a set of distinctive points which are referred to a global reference system. The main advantage of landmark-based maps is the compactness of their representation. By contrast, this kind of maps requires the existence of structures or objects that are distinguishable enough.
The map building problem can be solved by a single robot (Moutalier & Chatila, 1989), but it will be more efficiently managed if there is a team of robots, which collaborates in the construction of the map (Howard, 2006). In this case, the space can be divided so that the distances traversed are shorter and thus the odometry errors will be smaller. In this work,

we focus on this approach. In this context, two main proposals can be found in the literature. On the one hand, there are some solutions in which the estimate of the maps and trajectories is performed jointly (Fenwick et al., 2002; Gil et al., 2007; Thrun & Liu, 2004). In this case, there is a unique map, which is simultaneous built from the observations of the robots. In this way, the robots have a global notion of the unexplored areas so that the cooperative exploration can be improved. Moreover, in a feature-based SLAM, a landmark can be updated by different robots in such a way that the robots do not need to revisit a previously explored area in order to close the loop and reduce its uncertainty. However, the maintenance of this global map can be computationally expensive and the initial position of the robots should be known, which may not be possible in practice.

On the other hand, some approaches consider the case in which each robot builds a local map independently (Stewart et al., 2003; Zhou & Roumeliotis, 2006). Then, at some point the robots may decide to fuse their maps into a global one. In (Stewart et al., 2003), there is some point where the robots arrange to meet in. At that point, the robots can compute their relative positions and fuse their maps. One of the main advantages of using independent local maps, as explained in (Williams, 2001), is that the data association problem is improved. First, new observations should be only matched with a reduced number of landmarks in the map. Moreover, when these landmarks are fused into a global map, a more robust data association can be performed between the local maps. However, one of the drawbacks of this approach is dealing with the uncertainty of the local maps built by different robots when merging them.

The map fusion problem can be divided into two subproblems: the map alignment and the fusion of the data. The first stage consists in computing the transformation between the local maps, which have different reference systems. Next, after expressing all the landmarks in the same reference system, the data can be fused into a global map. In this work, we focus on the alignment problem in a multirobot visual SLAM context.

2. Map Building

The experiments have been carried out with Pioneer-P3AT robots, provided with a laser sensor and STH-MDCS2 stereo head from Videre Design. The stereo cameras have been previously calibrated and obtain 3D information from the environment. The maps thus built, are made of visual landmarks. These visual landmarks consist of the 3D position of the distinctive points extracted by the Harris Corner detector (Harris & Stephens, 1998). These points have an associated covariance matrix representing the uncertainty in the estimate of the landmarks. Furthermore these points are characterized by the U-SURF descriptor (Bay et al., 2006). The selection of the Harris Corner detector combined with the U-SURF descriptor is the result of a previous work, in which the aim was to find a suitable feature extractor for visual SLAM (Ballesta et al., 2007; Martinez Mozos et al., 2007; Gil et al., 2009).

The robots start at different positions and perform different trajectories in a 2D plane, sharing a common space in a typical office building. The maps are built with the FastSLAM algorithm using exclusively visual information. Laser readings are used as ground truth. The number of particles selected for the FastSLAM algorithm is M=200.

The alignment experiments have been initially carried out using two maps from two different robots (Section 5.1 and 5.2). Then, four different maps were used for the multi-robot alignment experiments (Section 5.2.1). The trajectories of the robots can be seen in

Figure 1. The laser measurements have been used as ground truth in order to estimate the accuracy of the results obtained.

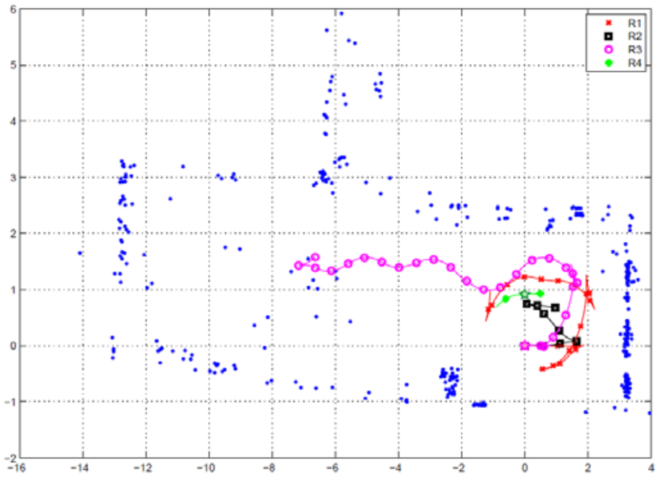


Fig. 1. Trajectories performed by four Pioneer P3AT robots and a 2D view of the global map.

3. Map alignment

The main objective of this work is to study the alignment stage in a multi-robot visual SLAM context. At the beginning, the robots start performing their navigation tasks independently, and build local maps. Given two of these feature maps, computing the alignment means computing the transformation, if existent, between those maps. In this way the landmarks belonging to different maps are expressed into the same reference system. Initially, before the alignment is performed, the local map of each robot is referred to its local reference system which is located at the starting point of the robot.

In order to compute the transformation between local maps, some approaches try to compute the relative poses of the robots. In this sense, the easiest case can be seen in (Thrun, 2001), where the relative pose of the robots is suppose to be known. A more challenging approach is presented in (Konolige et al., 2003; Zhou & Roumeliotis, 2006). In these cases, the robots, being in communication range, agree to meet at some point. If the meeting succeed, then the robots share information and compute their relative poses. Other approaches present feature-based techniques in order to align maps (Se et al., 2005; Thrun & Liu, 2004). The basis of these techniques is to find matches between the landmarks of the local maps and then to obtain the transformation between them. This paper focuses on the last approach.

In our case, although the maps are 3D, the alignment is performed in 2D. This is due to the fact that the robots' movements are performed in a 2D plane. The result of the alignment is a translation in x and y (t_x and t_y) and a rotation θ. This can be expressed as a transformation matrix T:

$$T = \begin{pmatrix} \cos\theta & \sin\theta & 0 & t_x \\ -\sin\theta & \cos\theta & 0 & t_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1}$$

Given two maps m and m′, T transforms the reference system of m′ into the reference system of m.

In order to select an adequate method to align this kind of maps, we have performed a comparative evaluation of a set of aligning methods (Section 4). All these methods try to establish correspondences between the local maps by means of the descriptor similarity. Furthermore, we have divided this study into to stages: first with simulated data (Section 5.1) and then with real data captured by the robots (Section 5.2). These experiments are performed between pairs of maps. However, we have additionally considered the multi-robot case, in which the number of robots is higher than 2. In this case, the alignment should be consistent, not only between pair of maps but also globally. This is explained in detail in the next section (Section 3.1).

3.1 Multi-robot alignment

This section tackles the problem in which there are n robots (n>2) whose maps should be aligned. In this case, the alignment should be consistent not only between pairs of maps but also globally. In order to deal with this situation, some constraints should be established (Se et al., 2005).

First, given n maps (n>2) and having each pair of them an overlapping part, the following constraint should be satisfied in the ideal case:

$$T_1 \cdot T_2 \cdot \ldots \cdot T_n = I \tag{2}$$

where I is a 3X3 identity matrix. Each T_i is the transformation matrix between map$_i$ and map$_{i+1}$ and corresponds to the matrix in Equation 1. The particular case of T_n refers to the transformation matrix between map$_n$ and map$_1$. The equation 2 leads to three expressions that should be minimized:

E1. $\sin(\theta_1+\ldots+\theta_n)$

E2. $t_{x1} + t_{x2}\cos(\theta_1) + t_{y2}\sin(\theta_1) + t_{x3}\cos(\theta_1+\theta_2) + t_{y3}\sin(\theta_1+\theta_2) + \ldots + t_{xn}\cos(\theta_1+\ldots+\theta_{n-1}) + t_{yn}\sin(\theta_1+\ldots+\theta_{n-1})$

E2. $t_{y1} + t_{x2}\sin(\theta_1) + t_{y2}\cos(\theta_1) - t_{x3}\sin(\theta_1+\theta_2) + t_{y3}\cos(\theta_1+\theta_2) + \ldots - t_{xn}\sin(\theta_1+\ldots+\theta_{n-1}) + t_{yn}\cos(\theta_1+\ldots+\theta_{n-1})$

Additionally, given a set of corresponding landmarks between map$_i$ and map$_{i+1}$, and having been aligned the landmarks of map$_{i+1}$ (L_j) into map$_1$'s coordinate system with the transformation matrix T_i (see Equation 1), the following expression should be minimized:

$$L_{Aj\{m\{k\}\}} - L_{i\{m(k)\}} \tag{3}$$

where m(k) is the total number of correspondences between the k-pair of maps ($k \in \{1,n\}$). The number of equations that emerge from Equation 3 is 2m(1)+2m(2)+…+2m(n). For instance, if we have m(1) common landmarks between map$_1$ and map$_2$ and the

transformation matrix between them is T_1, then for each common landmark we should minimize the following set of expressions:
$E\delta$. $x2\cos(\theta_1)+y2\sin(\theta_1)+tx1-x1$ with $\delta \in \{4,X+4\}$
$E\lambda$. $y2\cos(\theta_1)-x2\sin(\theta_1)+ty1-y1$ with $\lambda \in \{X+5,3X+5\}$
where $X=m(1)+m(2)+\ldots+m(n)$
So far, we have a non-linear system of $S = 3 + 2m(1) +\ldots+ 2m(n)$ constraints that we should minimize. In order to obtain the aligning parameters that minimize the previous S constraints, we used the **fsolve** MATLAB function. This iterative algorithm uses a subspace trust-region method which is based on the interior-reflective Newton method described in (Coleman, 1994; Coleman, 1996). The input for this algorithm is an initial estimate of the aligning parameters. This is obtained by the RANSAC algorithm of Sec. 4.1 between each pair of maps, i.e., map_1-map_2, map_2-map_3, map_3-map_4 and map_4-map_1. This will be the starting point of the results obtained with fsolve function to find a final solution.

4. Aligning methods

4.1 RANSAC (Random Sample Consensus)

This algorithm has been already aplied to the map alignment problem in (Se et al., 2005). The algorithm is performed as follows.
(a) In the first step, a list of possible corresponedences is obtained. The matching between landmarks of both maps in done based on the Euclidean distance between their associated descriptors d_i. This distance should be the minimum and below a threshold th_0 = 0.7. As a result of this first step, we obtain a list of matches consisting of the landmarks of one of the maps and their correspondences in the other map, i.e., m and m'.
(b) In a second step, two pair of correspondences ($[(x_i,y_i,z_i),(x_i',y_i',z_i')]$) are selected at random from the previous list. These pairs should satisfy the following geometric constraint (Se et al., 2005):

$$|(A^2 + B^2)-(C^2+D^2)| < th_1 \quad (4)$$

where $A = (x_i'-x_j')$, $B = (yi'-yj')$, $C = (xi-xj)$ and $D = (yi-yj)$. We have set the threshold th_1 = 0.8 m. The two pairs of correspondences are used to compute the alignment parameters (t_x, t_y, θ) with the following equations:

$$t_x= x_i - x_i'\cos\theta - y_i' \sin\theta \quad (5)$$

$$t_y= y_i - y_i'\cos\theta + x_i' \sin\theta \quad (6)$$

$$\theta = \arctan((BC-AD)/(AC+BD)) \quad (7)$$

(c) The third step consist in looking for correspondences that support the solution obtained (t_x, t_y, θ). Concretely, we transform the landmarks of the second map using the alignment obtained, so that it is referred to the same references system as the first map. Then, for each landmark of the transformed map, we find the closest landmark of the first map in terms of the Euclidean distance between their positions. The pairing is done if this distance is the

minimum and is below the threshold th_2=0.4m. As a result, we will have a set of matches that support the solution of the alignment.
(d) Finally, steps (c) and (d) are repeated M=70 times. The final solution will be that one with the highest number of supports.
In this algorithm, we have defined three different thresholds: th_0=0.7 for the selection of the initial correspondences, th_1=0.8 for the geometric constraint of Equation 4 and th_2=0.4m for selecting supports. Furthermore, a parameter min =20 establishes the minimum number of supports in order to validae a solution and M=70 is the number of times that steps (c) and (d) are repeated. These are considered as internal parameters of the algorithm and their values have been experimentally selected.

4.2 SVD (Singular Value Decomposition)

One of the applications of the Singular Value Decomposition (SVD) is the registration of 3D point sets (Arun et al., 1987; Rieger, 1987). The registration consists in obtaining a common reference frame by estimating the transformations between the datasets. In this work the SVD has been applied for the computation of the alignment between two maps. We first compute a list of correspondences. In order to construct this list (m, m′), we impose two different constraints. The first one is tested by performing the first step of the RANSAC algorithm (Section 4.1). In addition, the geometric constraint of Equation 4 is evaluated. Given this list of possible correspondences, our aim is to minimize the following expression:

$$||Tm'-m|| \qquad (8)$$

where m are the landmarks of one of the maps and m' their correspondences in the other map. On the other hand, T is the transformation matrix between both coordinate systems (Equation 1). T is computed as shown in Algorithm 1 of this section.

<u>Algorithm 1.</u> Computation of T, given m and m′

```
1: [u,d,v] = svd(m′)
2: $z=u^T m$
3: sv=diag(d)
4: $z_1$=z(1:n) {n is the Lumber of eigenvalues (not equal to 0) in sv}
5: w=$z_1$./sv
6: $T=(v*w)^T$
```

4.3 ICP (Iterated Closest Point)

The Iterative Closest Point (ICP) technique was introduced in (Besl & McKay, 1992; Zhang, 1992) and applied to the task of point registration. The ICP algorithm consists of two steps that are iterated:
(a) Compute correspondences (m, m′). Given an initial estimate T_0, a set of correspondences (m,m′) is computed, so that it supports the initial parameters T_0. T_0 is the transformation matrix between both maps and is computed with Equations 5, 6 and 7.
(b) Update transformation T. The previous set of correspondences is used to update the transformation T. The new T_{x+1} will minimize the expression: $||T_{x+1} \cdot m'-m||$, which is

analogous to the expression (5). For this reason, we have solved this step with the SVD algorithm (Algorithm 1).
The algorithm stops when the set of correspondences does not change in the first step, and therefore T_{x+1} is equal to T in the second step.
This technique needs an accurate initial estimation of the transformation parameters so that it converges properly. For that reason, in order to obtain an appropriate initial estimate we perform the two first steps in RANSAC algorithm (Section 4.1). The same threshold values are used.

4.4 ImpICP (Improved Iterated Closest Point)

The improved ICP (ImpICP) method is a modification of the ICP algorithm presented in Section 4.3, which has been implemented ad hoc. This new version is motivated by the importance of obtaining a precise initial estimation of the transformation parameter T_0. The accuracy of the results obtained is highly dependent on the goodness of this initial estimate. For that reason, in this new version of the ICP algorithm, we have increased the probability of obtaining a desirable result. Particularly, we obtain three different initial estimates instead of only one. This is performed by selecting three different pairs of correspondences each case in the second step of the RANSAC algorithm (Section 4.1), leading to three initial estimates. For each initial estimate, the algorithm runs as in Section 4.3. Finally, the solution selected is the transformation that is supported by a highest number of correspondences.

5. Experiments

The aim of this work is to find a suitable aligning method so that two or more maps can be expressed in the same reference system. This method should be appropriate for the kind of maps that our robots build, that is to say, landmark-based maps. With this idea, we have selected a set of algorithms that satisfy this requirement (See Section 4).
In order to perform these experiments, we have organised the work in two stages: first, a comparison of the aligning methods selected, using simulated data. In this case, we vary the amount of noise of the input data and observe the results by aligning pairs of maps. Secondly, we repeated the same experiments using real data captured by the robots. Furthermore, we include an experiment showing the performance of the multi-alignment case explained in Section 3.1, in which the number of maps we want to align is higher than 2.

5.1 Simulated Data

In order to perform the comparison between the aligning methods, we have built two 3D feature maps as can be seen in Figure 2. The coordinates of the landmarks have been simulated so that the alignment is evaluated with independence of the uncertainty in the estimate of the landmarks. Then, these points are described by U-SURF, extracted from real images which are typical scenarios of our laboratory. map_1 from Figure 2 has 250 points (stars), whereas map2 (circles) has a common area with map_1, whose size is variable, and a non-overlapping part which has 88 points. During the experimental performance, we test with different sizes of the overlapping area between these two maps (pentagons), so that we can observe the performance of the aligning methods vs. different levels of coincidence between the maps.

map_2 is 0.35 rads rotated and t_x = 5m, t_y = 10 m translated from map1. The size of the maps is 30X30 metres. Coincident points between the maps have initially the same descriptors. However, a Gaussian noise has been added to map2 so that the data are closer to reality. As a consequence, map2 has noise with σL in the localization of the points (coordinates' estimate) and noise with σD in the descriptors. The magnitude of σL and σD has been chosen experimentally.

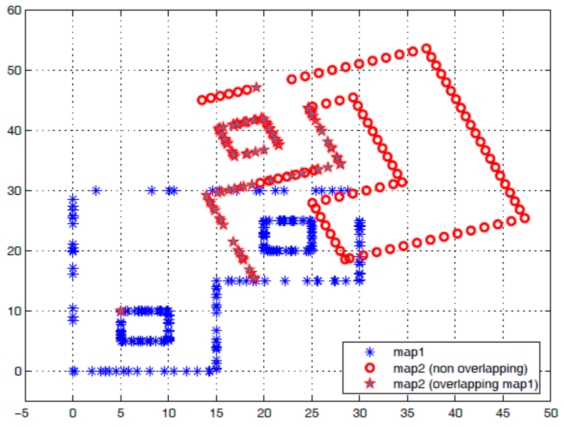


Fig. 2. 2D view of map1 and map2 (simulated data).

Figures 3, 4, 5 and 6 represent the results obtained with a noise of σL and σD equal to 0.20, whereas in Figures 7, 8, 9 and 10 these values are 0.50. In the X-axis, the number of points that both maps have in common is represented. This value varies from 0 to 160. The first value shows the case in which the maps do not overlap at all. For each value, the experiment is repeated 10 times. Then the Mean Quadratic Error is shown in the Y-axis (blue line). This value is computed comparing the alignment obtained and a ground truth. The blue points represent the individual values of the Error in each one of the 10 repetitions. In a similar way, the number of supports is also included in the graphics (red points). The number of supports is the number of correspondences that satisfy the transformation obtained. The mean value of supports is represented with a red line. In Figures 3 and 7, the green line represents the minimum value of supports required to validate the solution. Finally, all the figures present with bars the number of failures obtained in the 10 repetitions. Each failure represents the case in which the method does not converge to any solution or the solution does not satisfy the requirements (RANSAC method). In these cases, we consider that no alignment has been found.
Figures 3 and 7 show the results obtained with the RANSAC algorithm of Sec. 4.1. In figure 3, no solution is obtained until the number of overlapping points is higher than 60 points. In all of those cases, the Mean Quadratic Error always below 2 m. Regarding the number of supports (red line), we observe an ascendant tendency due to the increasing number of overlapping points. In the case of 160 overlapping points, the number of supports is 80. If the Gaussian noise is higher, these results get worse, as in Figure 7 where the number of supports obtained is significantly lower. Furthermore, the first solution appears when the number of overlapping points is 120.

Figures 4 and 8 present the results of the SVD algorithm of Section 4.2. In those cases, the error value having 100 overlapping points is close to 30. At least, the error has a descendent tendency as the number of overlapping points increases. However, in Fig 8 the error values are much more unstable. Regarding the number of supports, the tendency is quite constant in both graphics.

The behaviour of the ICP algorithm of Section 4.3 is presented in Figs. 5 and 9. In Figure 5 the error value obtained is quite acceptable. It is noticeable that the error curve decreases sharply from the case of 20 to 60 overlapping points. Then, the curve continues descending very slightly. This last part of the curve shows that the error values are around 2, which is a quite good result. Nevertheless, the yellow bars show, in some cases, a small number of failures. Fig. 9 shows worse results.

Finally, in Figures 6 and 10, the results of the improved version of ICP are shown. In this case, the results obtained are similar to that of ICP in terms of mean support values. However, it is noticeable that the stability of the algorithm is higher. If we pay attention to the yellow bars in Figure 6, it is shown that the algorithm always obtains a solution when the number of overlapping points is equal or higher than 100. In Figure 10, the number of failures is lower than in Figure 9.

In general, the best results are obtained by the ImpICP and RANSAC algorithms. RANSAC obtains lower error values, whereas ImpICP is more stable in terms of having less number of failures.

In addition to the experiments performed to evaluate the accuracy and suitability of the aligning methods, we have also measured the computational time of each algorithm (see Figure 11). The curves show an ascendant tendency. This is due to the fact that the size of map2 is higher as the number of overlapping points increases. It is remarkable that the values of the computation time are very similar in all of the methods. For that reason, this criterion can not be used to select one of the methods.

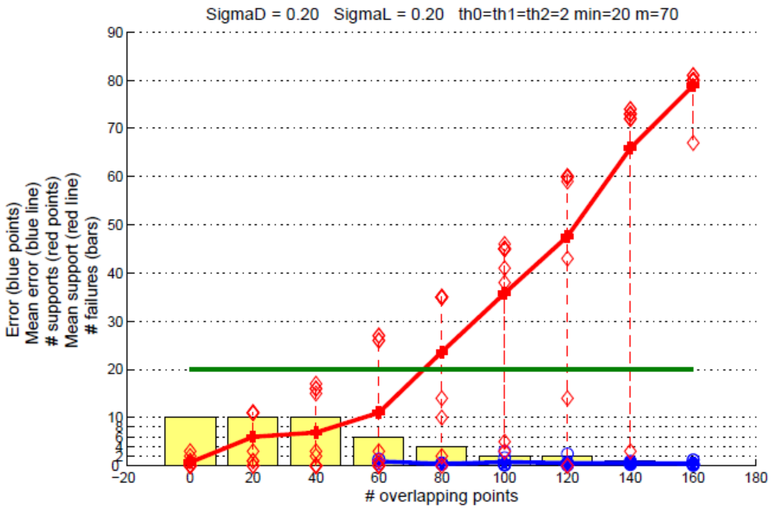


Fig. 3. RANSAC algorithm. The Gaussian noise is $\sigma_D = \sigma_L = 0.20$.

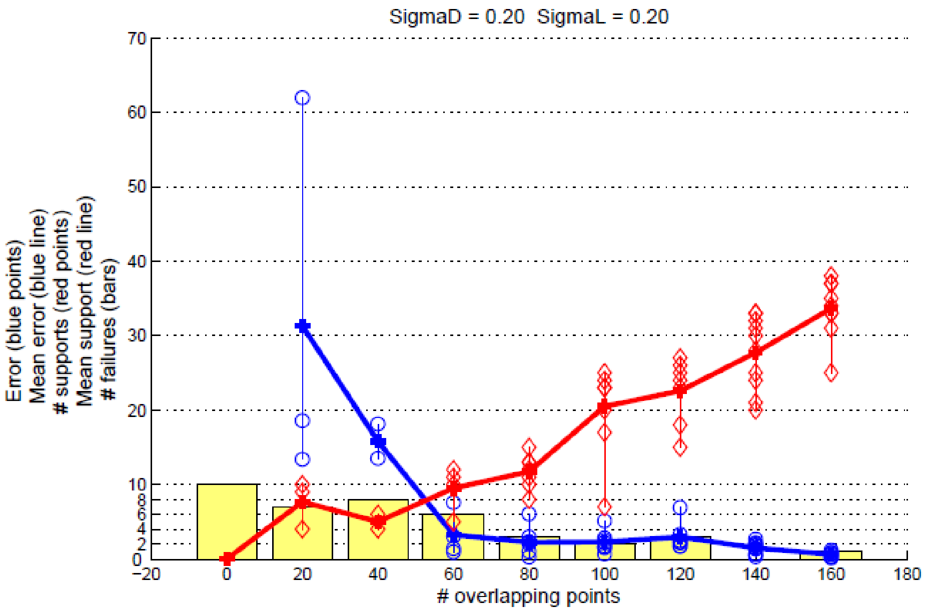


Fig. 4. ICP algorithm. The Gaussian noise is $\sigma_D = \sigma_L = 0.20$.

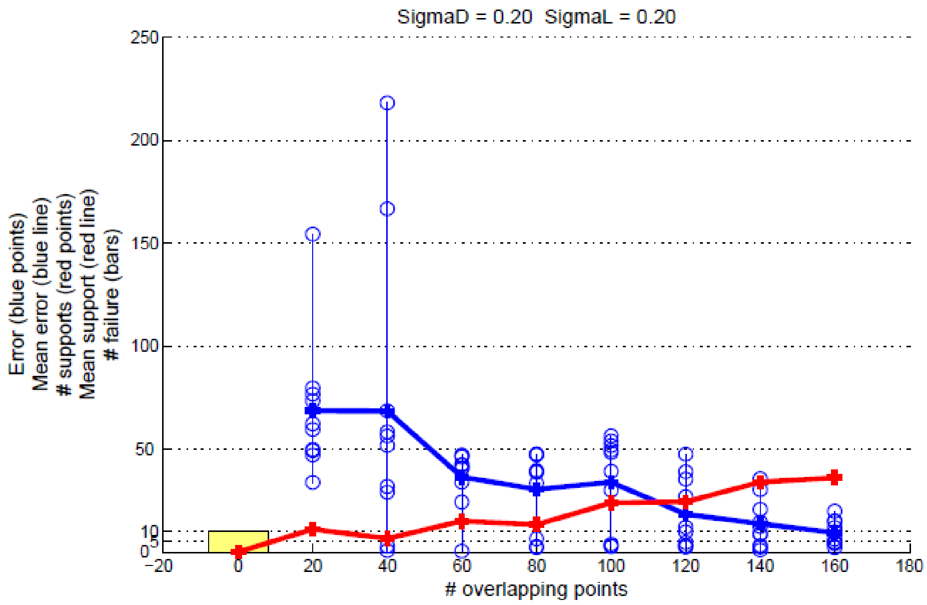


Fig. 5. SVD algorithm. The Gaussian noise is $\sigma_D = \sigma_L = 0.20$.

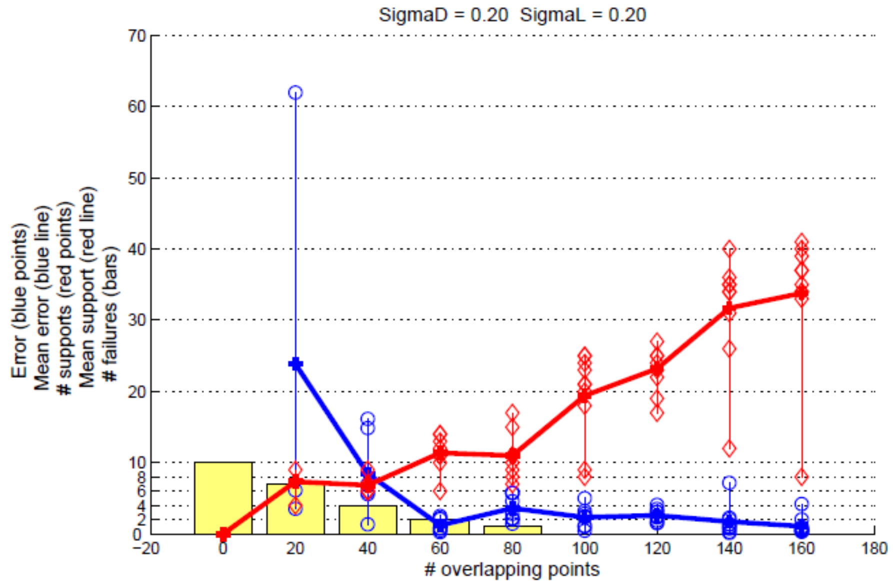


Fig. 6. ImpICP algorithm. The Gaussian noise is $\sigma_D = \sigma_L = 0.20$.

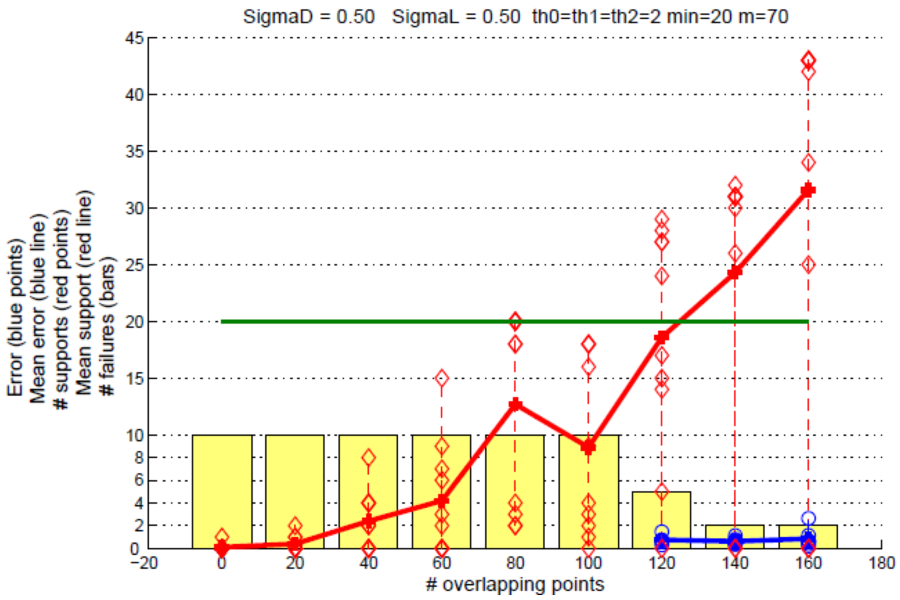


Fig. 7. RANSAC algorithm. The Gaussian noise is $\sigma_D = \sigma_L = 0.50$.

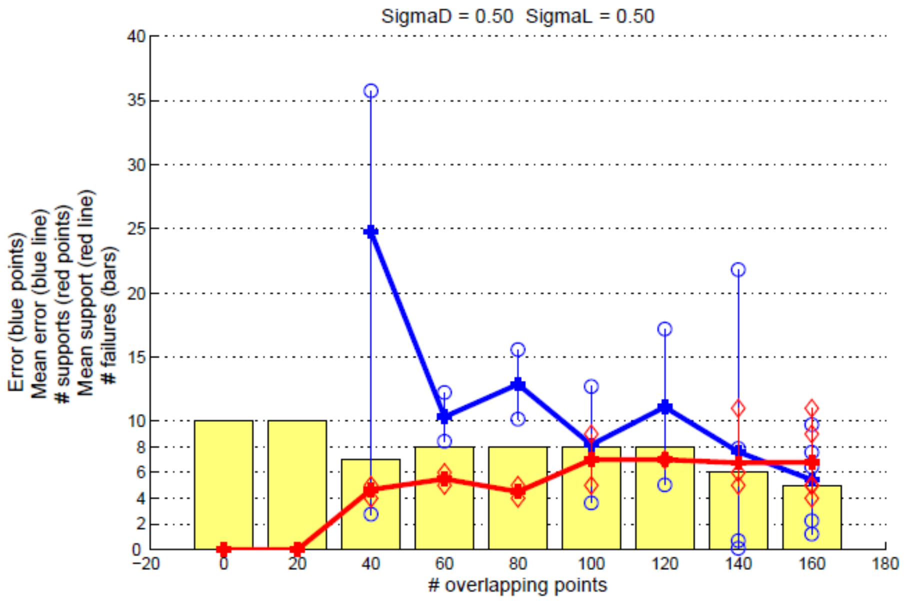


Fig. 8. ICP algorithm. The Gaussian noise is $\sigma_D = \sigma_L = 0.50$.

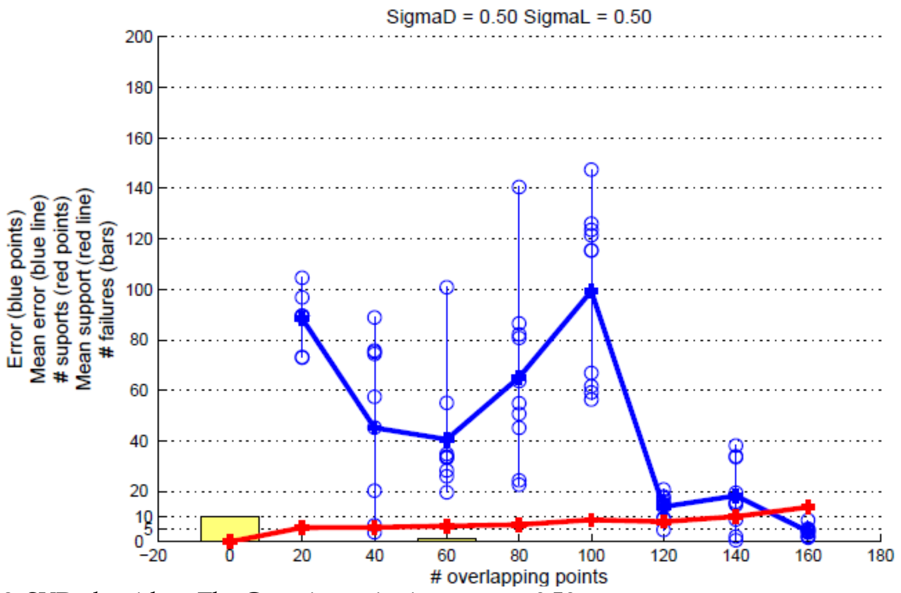


Fig. 9. SVD algorithm. The Gaussian noise is $\sigma_D = \sigma_L = 0.50$.

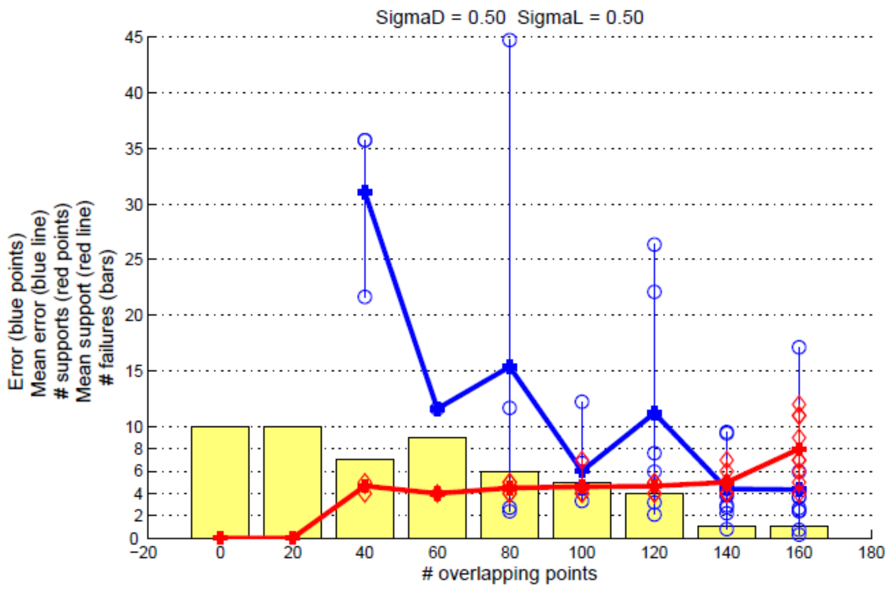


Fig. 10. ImpICP algorithm. The Gaussian noise is $\sigma_D = \sigma_L = 0.50$.

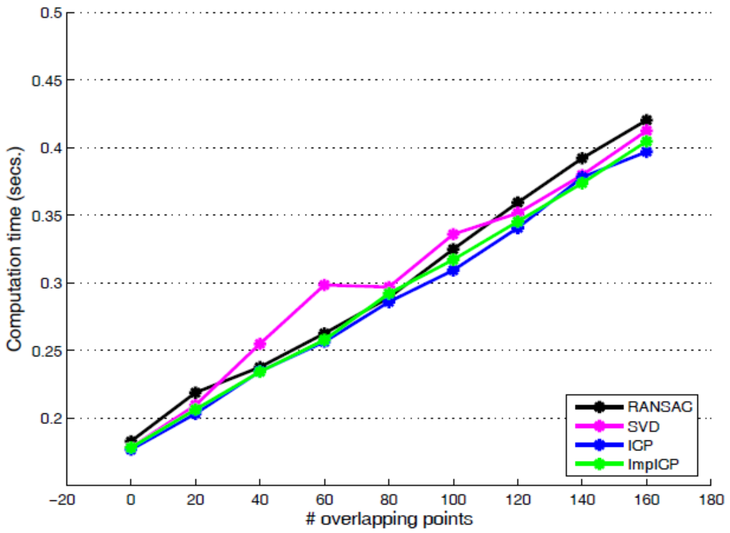


Fig. 11. Computational time vs. number of overlapping points.

5.2 Real Data

After performing the comparative analysis with the simulated data, the next step is to evaluate the same aligning methods using real data captured by the robots, i.e., landmarks consisting of Harris points detected from the environment and described by U-SURF.

We evaluate the performance of the aligning methods at different steps of the mapping process, i.e., at different iterations of the FastSLAM algorithm. At the beginning, the maps built by each robot have a reduced number of landmarks and therefore there are fewer possibilities of finding correspondences between these local maps. However, this situation changes as long as the maps are bigger. In this situation the probability of finding correspondences is higher and it is expected to obtain the alignment successfully.
In these experiments we have used the most probable map of each robot in order to compute the transformation between their reference systems. We obtain the most probable map of each robot at different iterations of the FastSLAM algorithm and try to align these maps. The most probable map is the map of the most probable particle of the filter in each particular moment.
The FastSLAM algorithm is performed in several iterations corresponding to the total number of movements performed by the robot. In the experiments k is an index that denotes the number of iterations. In this case, this number is k=1410 and the sizes of the maps at that point are map_1=263 landmarks and map_2=346 landmarks. These maps have a dimension of 35X15 meters approximately.
In Figure 15(a), we can observe a 2D view of the local maps constructed by each robot and referred to its local frame. In this figure, map_1 is represented by stars and has 181 landmarks. On the other hand, map_2 is represented by circles and it size is of 187 landmarks. In Figure 15(b), we can see the two local maps already aligned. In this case, the most probable maps of iteration k=810 have been used.
In order to compare the aligning methods with real data, we compute the alignment parameters for each method at different iterations of the FastSLAM algorithm. The evaluating measure is the Euclidean distance between the alignment parameters tx,ty and θ the real relative position between the robots, denoted as ground truth. This measure was obtained estimating the relative position of the robots being at their starting positions.
Figure 12, illustrates the comparison of the aligning methods we evaluate. For each method, the error values (y axis) vs. the k-iteration of the algorithm (x axis) are represented. Logically, as the number of iterations increases, the size of the maps constructed will be higher and therefore it will be more probable to find a solution closer to the ground truth. For this reason, it is expected to obtain small error values as the k-iteration increases. In Figure 12 we can observe that the worst results are obtained with SVD. For instance, SVD has an error of 4m with k-iteration =1409, i.e., at the end of the FasSLAM algorithm. Next, ICP obtains similar results. However, it achieves better results in some cases. For example, with k-iteration = 810 the error is lower than 1 m. Then, the ImpICP algorithm outperforms these previous methods, since it achieves really small error values. Nevertheless, RANSAC is the method that obtains better results. Despite the fact that it gives no solution with k-iteration = 60 (probably because the maps are still too sparse in this iteration), the algorithm obtains the smallest error values. In fact, from k-iteration =410 on the error is no higher than 0.5m. Finally, Figures 13 and 14 focus on the results obtained by the RANSAC algorithm. Figure 13 shows the number of supports obtained in each case, which increases with the k-iteration values. On the other hand, Figure 14 shows the decomposition of the error in its three components (three alignment parameteres): error in tx, ty and θ.

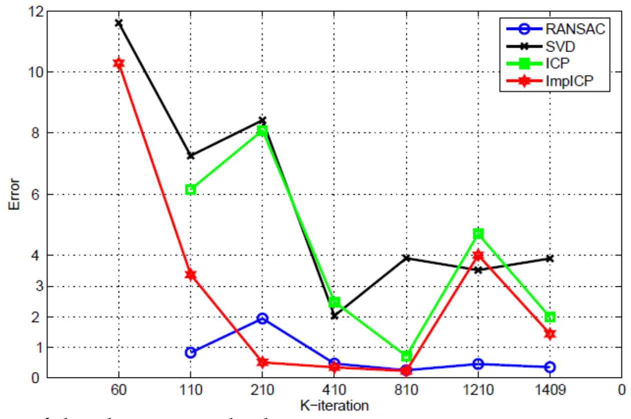


Fig. 12. Evaluation of the aligning methods.

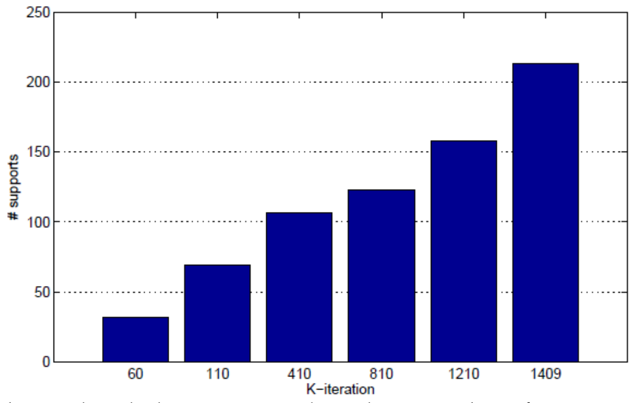


Fig. 13. Results obtained with the RANSAC algorithm. Number of supports.

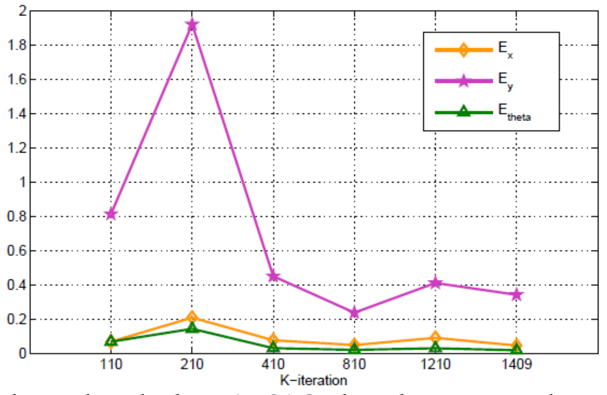


Fig. 14. Results obtained with the RANSAC algorithm. Error values for each aligning parameter.

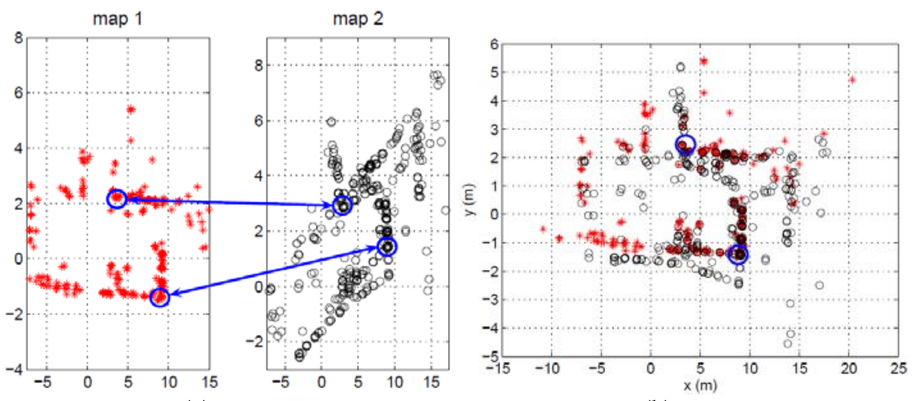


Fig. 15. Map alignment with real data. (a) Local maps before de alignment. Example of detected correspondences. (b) Map after the alignment.

5.2.1 Multi-alignment results.

Table 1 presents an example of the results obtained with the **fsolve** function (Section 3.1). This table shows the aligning results obtained in the group of four robots, where T^i_j represents the alignment between robot i and robot j. On the top part of the table, we can observe the aligning results between each pair of robots. These alignment parameters (t_x,t_y and θ) have been computed by means of the RANSAC algorithm described in Section 4.1. These solutions are valid between pairs of maps but may not be consistent globally. Then, on the bottom of the table, the alignment parameters (t'_x,t'_y and θ′) have been obtained with the fsolve function. In this case, the constraints imposed (see expressions E1 to Eλ of Section 3.1) optimize the solution so that it is globally consistent.

	T^1_2	T^2_3	T^3_4	T^4_1
tx	-0.0676	0.1174	-0.0386	0.0547
ty	-0.0636	0.0423	0.8602	-0.8713
θ	-0.0144	-0.0063	0.0286	-0.0248
tx'	-0.0388	0.0677	-0.0408	0.0774
ty'	0.0363	-0.1209	0.9521	-0.9220
θ	0.0079	-0.0375	0.0534	-0.0436

Table 1. Alignment parameters.

7. Conclusion

This work has been focussed on the alignment problem of visual landmark-based maps built by the robots. The scenario presented is that of a team of robots that start their navigation tasks from different positions and independently, i.e., without knowledge of other robots' positions or observations. These robots share a common area of a typical office building. The maps built with the FastSLAM algorithm are initially referred to the reference system of each robot, located at their starting positions. In this situation, we consider the possibility of merging these maps into a global map. However, in order to do that, the landmarks of these

maps should be expressed in the same reference system. This is the motivation for the study of the alignment problem. To do that we have perform a comparison of several algorithms in order to select the most suitable for this kind of maps. The experiments have been carried out using simulated data as well as real data captured by the robots. The maps built by the robots are 3D maps. Nevertheless the alignment is performed in 2D, since the movement of the robots is performed in a 2D plane.
The next step is the study the second stage: map merging, i.e., fusing all the data into a global map. In this case, the uncertainty in the estimate of the landmarks should be considered. The map fusion problem has to be conceived as integrated into the FastSLAM problem, in which each robot pose is represented by a set of particles and each of them have a different estimate of the map. With this idea, observations coming from other robots should be treated different in terms of uncertainty. Furthermore, some questions are still open such as: when do we fuse the maps, how do we use the global map after the fusion is performed, etc. These ideas will be consider as future work.

8. Acknowledgment

This work has been supported by the Spanish Government under project 'Sistemas de Percepción Visual Móvil y Cooperativo como Soporte para la Realización de Tareas con Redes de Robots' CICYT DPI2007-61107 and the Generalitat Valenciana under grant BFPI/2007/096.

9. References

Arun, K.S. ; Huang, T. S. & Blostein, S.D. (1987). Least square fitting of two 3d sets. *IEEE Transactions on Patterns Analysis and Machine Intelligence.* Vol. PAMI-9, No. 5, pp. 698-700, ISSN:0162-8828.

Ballesta, M. ; Gil, A. ; Martinez Mozos, O. & Reinoso, O. (2007). Local descriptors for visual SLAM. *Workshop on Robotics and Mathematics (ROBOMAT07)*, Portugal.

Bay, H. ; Tuytelaars, T. & Van Gool, L. (2006). SURF: Speeded-up robust features. *Proceedings of the 9th European Conference on Computer Vision,* 2006, pp. 404-417.

Besl, P.J. & Mckay, N. (1992). A method for registration of 3-d shapes. *IEEE Transactions on Patterns Analysis and Machine Intelligence.* Vol. PAMI-14, No. 2, pp. 239-256.

Coleman, T.F. (1994). On the convergence of reflective newton methods for largerscale nonlinear minimization subject to bounds. *Mathematical Programming*. Vol. 67, No. 2, pp. 189-224.

Coleman, T.F. (1996). An interior, trust region approach for nonlinear minimization subject ot bounds. *SIAM Journal on Optimization*.

Fenwick, J.W. ; Newman, P.N. & Leornard, J. (2002). Cooperative concurrent mapping and localization. *Proceedings of the IEEE International Conference on Intelligent Robotics and Automation,* pp. 1810-1817, ISBN: 0-7803-7272-7, Washington, DC, USA, May 2002.

Gil, A. ; Reinoso, O.; Payá, L. & Ballesta, M. (2007). Influencia de los parámetros de un filtro de partículas en la solución al problema de SLAM. *IEEE Latin America*, Vol.6, No.1, pp.18-27, ISSN: 1548-0992.

Gil, A. ; Martinez Mozos, O. ; Ballesta, M. & Reinoso, O. (2009). A comparative evaluation of interest point detectors and local descriptors for visual slam. Ed. Springer Berlin / Heidelberg, ISSN: 0932-8092, pp. 1432-1769.

Howard, A. (2006). Multi-robot Simultanous Localization and Mapping using particle filters. *International Journal of Robotics Research,* Vol. 5, No. 12, pp. 1243-1256.

Kwak, N. ; Kim, G-W. ; Ji, S-H. & Bee, B-H. (2008). A mobile robot exploration strategy with low cost sonar and tungsten-halonen structural light. *Journal of Intelligent and Robotic Systems,* 5(1) : 89-111.

Konolige, K. ; Fox, D. ; Limketkai, B. ; Ko, J. & Stewart, B. (2003). Map merging for distributed robot navigation. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robotics and Systems.*

Leonard, J.J. & Durrant-Whyte, H.F. (1991). Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation,* Vol. 7, No. 3, pp. 376-382, ISSN 1042-296X.

Martínez Mozos, O. ; Gil, A. ; Ballesta, M. & Reinoso, O. (2007). Interest point detectors for visual SLAM. *Proceedings of the XII Conference of the Spanish Association ofr Artificial Intelligence (CAEPIA),* Salamanca, Spain.

Montemerlo, M.; Thrun, S.; Koller, D. & Wegbreit, B. (2002). FastSLAM: A factored solution to simultaneous localization and mapping. *Proceedings of the National Conference on Artificial Intelligence (AAAI).* Edmonton, Canada, pp. 593-598.

Moutalier, P. & Chatila, R. (1989). An experimental system for incremental environment modeling by an autonomous mobile robot. *1st International Symposium on Experimental Robotics*. Montreal, Vol. 139/1990, ISSN 0170-8643.

Rieger, J. (1987). On the classification of views of piecewise smooth objects. *Image and Vision Computing.* Vol. 5, No. 2 (May 1987), pp. 91-97.

Se, S. ; Lowe, D. & Little, J.J. (2005). Vision-based global localization and mapping for mobile robots. *IEEE Transactions on robotics.* Vol. 21, No. 3, pp. 364-375, ISSN 1552-3098 .

Stewart, B. ; Ko, J. ; Fox, D. & Konolige, K. (2003). A hierarchical bayesian approach to mobile robot map structure estimation. *Proceedings of the Conference on Uncertainty in AI (UAI).*

Thrun, S. (2001). A probabilistic online mapping algorithm for teams of mobile robots. *International Journal of Robotics Research*. 20(5), pp. 335-363.

Thrun, S. & Liu,Y. (2004). Simultaneous localization and Mapping with sparse extended information filters. *International Journal of Robotics Research*. Vol. 23, No.7-8, pp. 693-716 (2004), ISSN 0278-3649.

Triebel, R. & Burgard, W. (2005). Improving simultaneous mapping and localization. *Proceedings of the National Conference on Artificial intelligence (AAAI).*

Valls Miro, J. ; Zhou, W. & Dissanayake, G. (2006). Towards vision based navigation in large indoor environments. *IEEE/RSJ Int. Conference on Intelligent Robots & Systems.*

Williams, S. (2001). Phd dissertation: Efficient solutions to autonomous mapping and navigation problems. Australian Center for Field Robotics, University of Sidney, 2001.

Wijk, O. & Christensen, H.I. (2000). Localization and navigation of a mobile robot using natural point landmark extracted from sonar data. *Robotics and Autonomous Systems*. 1(31), pp. 31-42.

Zhang, Z. (1992). On local matching of free-form curves. *Proceedings of BMVC.* Pp. 347-356.

Zhou, X .S. & Roumeliotis, S.I. (2006). Multi-robot slam with unknown initial correspondence: The robot rendezvous case. *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems,* Beijing, China, pp. 1785-1792, 2006.

Key Elements for Motion Planning Algorithms

Antonio Benitez, Ignacio Huitzil, Daniel Vallejo,
Jorge de la Calleja and Ma. Auxilio Medina
Universidad Politécnica de Puebla,
Universidad de las Américas – Puebla, México

1. Introduction

Planning a collision-free path for a rigid or articulated robot to move from an initial to a final configuration in a static environment is a central problem in robotics and has been extensively addressed over the last. The complexity of the problem is NP-hard (Latombe, 1991). There exist several family sets of variations of the basic problem, that consider flexible robots, and where robots can modify the environment. The problem is well known in other domains, such as planning for graphics and simulation (Koga et al., 1994), planning for virtual prototyping (Chang & Li, 1995), and planning for medical (Tombropoulos et al., 1999) and pharmaceutical (Finn & Kavraki, 1999) applications.

1.1 Definitions and Terminology

A robot is defined into the motion planning problems as an object, which is capable to move (rotating and translating) in an environment (the workspace) and it may take different forms, it can be: a rigid object, an articulated arm, or a more complex form like a car or an humanoid form.

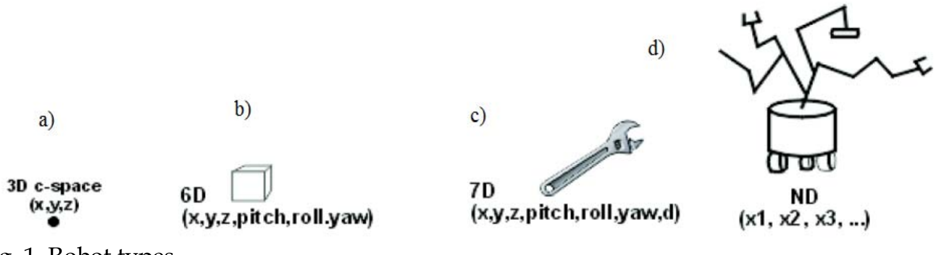


Fig. 1. Robot types.

Given different robot types, it is fully and succinctly useful to represent the position of every point of the robot in a given moment. As shown in Figure 1. (a), a robot can be represented as a point. When the robot is a point, as it is the case in theoretical discussion, it can be

completely described by its translational coordinates, (x, y, z). For a robot which is a rigid body freely moving in a 3D space (See Figure 1. (b)), the position of every point is represented by six parameters (x, y, z) for the position and (α, β, γ) for its rotation in every point on the space. Each parameter, or coordinate, necessary to give the full description of the robot, is called a degree of freedom (DOF). The seven DOF shown in Figure 1. (c) is a spanning wrench. It has the same six DOF as the cube robot plus a seventh DOF, the width of the tool jaw. The far right n DOF in Figure 1. (d) demonstrates that the number of DOF can become extremely large as robots become more and more complex.
Not only the number, but the interpretation of each DOF is necessary to fully understand the robot's position and orientation. Figure 2 (a) Shows six DOF that are necessary to describe a rigid object moving freely in 3D. Six parameters are also necessary to describe a planar robot with a fixed base and six serial links Figure 2. (b). Although the same size, the coordinate vectors of each robot are interpreted differently.

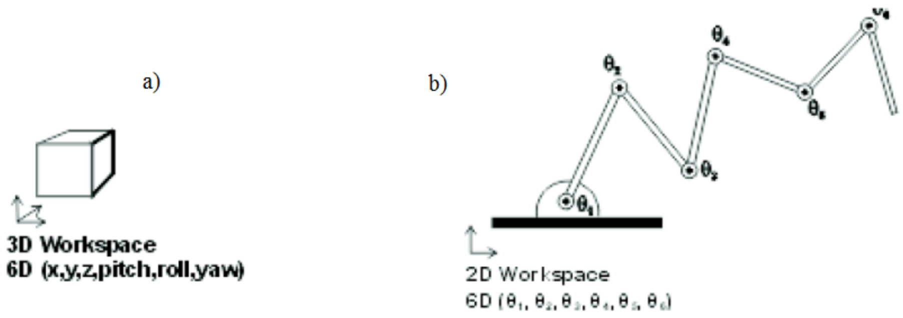

a)
3D Workspace
6D (x,y,z,pitch,roll,yaw)
b)
θ_1
θ_2
θ_3
θ_4
θ_5
θ_6
2D Workspace
6D (θ_1, θ_2, θ_3, θ_4, θ_5, θ_6)

Fig. 2. Number of Degrees of freedom for robots.

1.2 Paths and Connectivity

Throughout the history of artificial intelligence research, the distinction between *problem solving and planning* has been rather elusive (Steven, 2004). For example, (Russell & Norvig, 2003) devotes a through analysis of "Problem-solving" and "Planning". The core of the motion planning problem is to determine whether "a point of the space" is reachable from an initial one by applying a sequence of actions (Russell & Norvig, 2003), p. 375.
Besides, it is difficult to apply results from computational geometry to robot motion planning, since practical aspects relevant to robotics are often ignored in computational geometry, e.g. only static environments are considered usually. Since testing for collisions between the robot and the environment is essential to motion planning, the representation of geometry is an important issue.
The motion planning problem can be defined (Steven, 2004) as a continuum of actions that can lead to a path in a state space. The path is obtained through the integration of a vector field computed by using the plan. Thus, the plane has an important role because it describes a set of states.

1.3 Workspace and Configuration Space

The robot moves in a workspace, consisting of several objects, guided by natural lows. For motion planning algorithms, the concept of workspace can be defined by considering two or three dimensions; of course it can be defined by N-dimensions. In this context, the workspace consists of rigid objects (obstacles) with six DOF. Initially, obstacles are placed in static configurations, so they can not move within the environment. The workspace representation is associated to a geometric model used to manipulate the objects. Both features must be considered to address the motion planning problem. However, in many instances, it may be possible to improve performance by carefully investigating the constraints that arise for particular problems once again. It may be possible to optimize performance of some of the sampling-based planners in particular contexts by carefully considering what information is available directly from the workspace constraints.

1.3.1 Configuration Space

If the robot has *n* degrees of freedom, this leads to a manifold of dimension *n* called the *configuration space* or *C-space*. It will be generally denoted by C. In the context of this document, the configuration space may be considered as a special form of state space. To solve a motion planning problem, algorithms must conduct a search in this space. The configuration space notion provides a powerful abstraction that converts the complicated models and transformations into the general problem of computing a path in a manifold (Steven, 2004).

1.3.2 The motion planning problem

The basic motion planning problem is defined as follows: Given a robot, which can be any moving object, a complete description of static workspace and star and goal configurations, the objective is to find a valid (i.e., collision free) path for the robot to move through the workspace from beginning to goal configurations. The robot must avoid collision with itself as well as obstacles in the workspace environment.

1.3.3 Probabilistic roadmap methods

A class of motion planning methods, known as probabilistic roadmap methods (*PRMs*), have been largely addressed (Ahuactzin, & Gupta, 1997), (Amato et al., 1998), (Boor et al., 1999). Briefly, *PRMs* use randomization to construct a graph (*a roadmap*) in configuration-space (C-space). *PRMs* provide heuristics for sampling C-space and C-obstacles without explicitly calculating either.

When PRM maps are built, roadmap nodes correspond to collision-free configurations of the robot, i.e. points in the free C-space (C-free). Two nodes are connected by an edge if a collision-free path between the two corresponding configurations can be found by a “local planning” method. Local planners take as input a pair of configurations and check the path (edge) between them for collision. As output they declare the path valid (collision-free) or invalid. Because of the large number of calls made to local planners, their design often sacrifices sophistication for speed. When local planners (Amato et al., 1998) are deterministic, the edges do not need to be saved, only the roadmap adjacency graph must saved. *PRM* methods may include one or more ‘enhancement’ steps in which some areas are

sampled more intensively either before or after the connection phase. The process is repeated as long as connections are found or a threshold is reached.
A particular characteristic of *PRM* is that queries are processed by connecting the initial and goal configurations to the roadmap, and then searching for a path in the roadmap between these two connection points.
The following pseudo-code summarizes the high-level algorithm steps for both roadmap construction and usage (query processing). Lower-level resources include distance metrics for evaluating the promise of various edges local planners for validating proposed edges, and collision detectors for distinguishing between valid (collision free) and invalid (in collision) robot configurations.

```
Pre-processing: Roadmap Construction
1. Node Generation (find collision-free configurations)
2. Connection (connect nodes to form roadmap)

(Repeat the node generation as desired)

On Line: Query Processing

1. Connect start/goal to roadmap
2. Find roadmap path between connection nodes

(repeat for all start/goal pairs of interest)
```

1.4 The narrow corridor problem

A narrow passage occurs when in order to connect two configurations a point from a very small set must be generated. This problem occurs independently of the combinatorial complexity of the problem instance. There have been several variants proposed to the basic *PRM* method which do address the “narrow passage problem”. Workspaces are difficult to handle when they are “cluttered”. In general, the clutter is made up of closely positioned workspace obstacles. Identifying “difficult” regions is a topic of debate. Nevertheless when such regions are identified the roadmap can be enhanced by applying additional sampling. Naturally, many of the nodes generated in a difficult area will be in collision with whatever obstacles are making that area difficult. Some researchers, not wishing to waste computation invested in generating nodes discovered to be in-collision, are considering how to “salvage” such nodes by transforming them in various ways until they become collision-free.

2. Probabilistic Roadmap Methods

This section presents fundamental concepts about PRM, including a complete description of PRM, OBPRM, Visibility Roadmap, RRT and Elastic Band algorithms. These methods are important because they are the underlying layer of this topic. The parametric concept of configuration space and its importance to build the roadmap is also presented.

The world (Workspace or W) generally contains two kinds of entities:

1. *Obstacles*: Portions of the world that are "permanently" occupied, for example, as in the walls of a building.

2. *Robots*: Geometric bodies that are controllable via a motion plan.

The dimension of the workspace determines the particular characteristic of W. Formulating and solving motion planning problems requires defining and manipulating complicated geometric models of a system of bodies in space. Because physical objects define spatial distributions in 3D-space, geometric representations and computations play an important role in robotics.

There are four major representation schemata for modelling solids in the physical space (Christoph, 1997). They are the follows. In constructive solid geometry (CSG) the objects are represented by unions, intersections, or differences of primitive solids. The boundary representation (BRep) defines objects by quilts of vertices, edges, and faces. If the object is decomposed into a set of nonintersecting primitives we speak of spatial subdivision. Finally, the medial surface transformation is a closure of the locus of the centres of maximal inscribed spheres, and a function giving the minimal distance to the solid boundary. We describe the boundary representation because this is related to our work.

2.1 Geometric Modelling

There exists a wide variety of approaches and techniques for geometric modelling. Most solid models use BRep and there are many methods for converting other schemata into BRep (Christoph, 1997). Research has focused on algorithms for computing convex hulls, intersecting convex polygons and polyhedron, intersecting half-spaces, decomposing polygons, and the closest-point problem. And the particular choice usually depends on the application and the difficulty of the problem. In most cases, such models can be classified as: 1) a boundary representation, and 2) a solid representation.

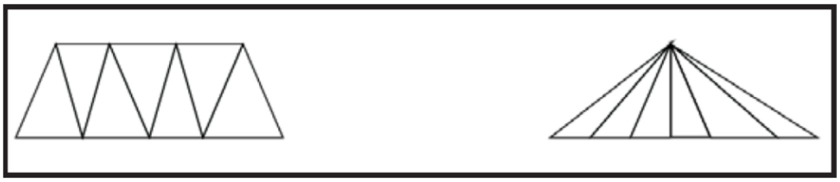


Fig. 3. Triangle strips and triangle fans can reduce the number of redundant points.

Suppose W = $\mathbb{R}^3$. One of the most convenient models to express the elements in W is a set of triangles, each of which is specified by three points, (x_1, y_1, z_1), (x_2, y_2, z_2) and (x_3, y_3, z_3). It is assumed that the interior of the triangle is part of the model. Thus, two triangles are considered as *colliding* if one pokes into the interior of another. This model is flexible because there are no constraints on the way in which triangles must be expressed. However, there exists redundancy to specify the points. Representations that remove this redundancy are triangle strips. *Triangle strips* is a sequence of triangles such that each adjacent pair shares. *Triangle fan* is triangle strip in which all triangles share a common vertex, as shown in Figure 3.

2.2 Rigid body transformations

Once we have defined the geometric model used to represent the objects in the workspace, it is necessary to know how these objects are going to be manipulated (the robot as a particular case) through the workspace. Let O refer to the obstacle region, which is a subset of W. Let A refer to the robot, which is a subset of $\mathbb{R}^2$ 2 or $\mathbb{R}^3$, matching the dimension of W. Although O remains fixed in the world, W, motion planning problems will require "moving" the robot, A.
Let A be a rigid body which we want to translate by some x_t, y_t, $z_t \in \mathbb{R}$ by mapping every $(x, y, z) \in A$ to $(x + x_t, y + y_t, z + z_t)$. Primitives of the form $H_i = (x, y, z) \in W \mid f_i(x, y, z) \leq 0$, are transformed to $(x, y, z) \in W \mid f_i(x - x_t, y - y_t, z - z_t) \leq 0$. The translated robot is denoted as $A(x_t, y_t, z_t)$. Note that a 3D body can be independently rotated around three orthogonal axes, as shown in Figure 4.

1. A *yaw* is a counter clockwise rotation of α about the Z-axis. The rotation is given by the following matrix.

$$R_Z(\alpha)=\begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (1)$$

Note that the upper left entries of $R_Z(\alpha)$ form a 2D rotation applied to the XY coordinates, while the Z coordinate remains constant.

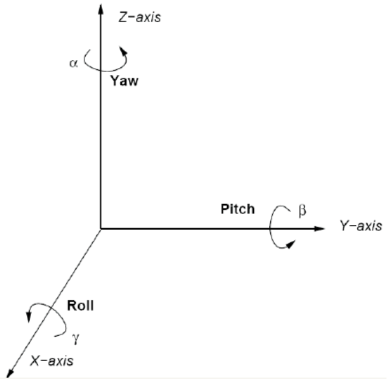


Fig. 4. Any rotation in 3D can be described as a sequence of yaw, pitch, and roll rotations.

2. A *pitch* is a counter clockwise rotation of β about the Y-axis. The rotation is given by the following matrix.

$$\begin{pmatrix} \cos\beta & 0 & \sin\beta \\ & & \\ & & \end{pmatrix} \quad (2)$$

$$R_Y(\beta)= \begin{matrix} 0 & 1 & 0 \\ -\sin\beta & 0 & \text{Cos}\,\beta \end{matrix}$$

3. A *roll* is a counter clockwise rotation of γ about the X-axis. The rotation is given by the following matrix.

$$R_x(\gamma)= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{pmatrix} \qquad (3)$$

As in the 2D case, a homogeneous transformation matrix can be defined. For the 3D case, a 4 X 4 matrix is obtained that performs the rotation given by R(α, β, γ), followed by a translation given by x_t, y_t, z_t. The result is defined as:

$$T= \begin{pmatrix} cos\alpha cos\beta & cos\alpha sin\,\beta\,cos\,\gamma - sin\alpha\,cos\,\gamma & cos\alpha\,sin\,\beta\,cos\,\gamma + sin\alpha\,sin\,\gamma & x_t \\ cos\beta & sin\alpha\,sin\beta sin\gamma + cos\alpha\,cos\,\gamma & sin\alpha\,sin\beta\,cos\,\gamma - cos\alpha\,sin\,\gamma & y_t \\ \beta & cos\beta sin\,\gamma & cos\beta cos\,\gamma & z_t \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (4)$$

The triangle meshes associated to the robot (using a triangle as primitive) can be transformed, to yield A(xt, yt, zt, α, β, γ).

2.3 Configuration Space

Configuration-space (C-space) is an abstraction of the motion planning problem. Briefly, the motion planning problem is expressed in a *n-dimensional* space, where *n* represents the number of DOF of the robot, and a robot configuration (all the necessary DOF's for fully describing the robot's position and pose) is a point. The C-space consist ALL possible points, those of free collision configuration corresponding to the robot free space and collision configuration corresponding to the robot in collision with or more obstacles or itself. In a 3D workspace, the path between any starting and configuration is a swept volume. In a given C-space, the same path is a one-dimensional curve traced by the C-space point representing the robot moving from a start to configuration. Only when the robot is *really* a point robot (as in theoretical discussion) the workspace and the robot's C-space are the same.
If the robot has *n* degrees of freedom, this leads to a manifold of dimension the *configuration space* or *C-space*. It will be generally denoted by C. Configurations space has different properties, next paragraphs describes them.

2.3.1 Paths

Let X be a topological space, which is a manifold. A *path*, τ , in X is a continuous function, τ : [0, 1] $\rightarrow X$. Other intervals of $\mathbb{R}$ may alternatively be used for the domain of τ . Note that a path is a function, not a set of points. Each point along the path is given by τ (s) for some $s \in$

[0, 1]. Recall in that case, a countable set of stages was denned, and the states visited could be represented as x_1, x_2,.... In the current setting τ (s) is used, in which s replaces the stage index. To make connection clearer, we could use x instead of τ, to obtain $x(s)$ for each $s \in$ [0, 1].

3. Traditional Roadmap Methods

A class of motion planning methods, known as probabilistic roadmap methods (*PRMs*), have made large recent gains in popularity. In general roadmap methods solve the motion planning problem in two phases: roadmap construction and roadmap querying (runtime). It is important to understand that the objective of the pre-processing phase is to build an adequate roadmap for any particular problem. Such result is used to quickly satisfy (typically within a few seconds) a large number of different queries to the same roadmap at execution time. It is not assumed that any crucial start and goal configurations are known a *priori* but rather that the entire workspace must be explored and mapped before the robot can efficiently operate there. The process is analogous to that of cartographers making a map of a country's road system. When they finish, many drivers can all obtain the same map and use it for their individual navigation purposes. There is no single start to goal; rather all possible routes must be explored and charted.

3.1 General PRM

Briefly, PRMs use randomization to construct a graph (a roadmap) in a configuration space (C-space) taking into account that there exist forbidden and feasible spaces. Therefore the algorithm computes configurations in both regions and tests for collision to determine which configurations are going to be retained. Once the configuration space has been sampled using a tool called "Local Planner". This local planner verifies if there exists a collision free path between two corresponding configurations. The path exists if two nodes are directly or transitively connected by an edge. As output the local planner declares a valid path (collision-free) or invalid.
Finally, queries are processed by connecting the initial and goal configurations to the roadmap, and then searching for a path in the roadmap between these two connection points, see Figure 5. As described (Kavraki& Latombe, 1994), (Kavraki et al., 1996), (Overmars & Svestka, 1994) the basic *PRM* uses uniform sampling to generate, uniformly and randomly configurations (nodes) of the robot in the workspace. The local planner that is likely to return failure by submitting only pairs of configurations whose relative distance (according to the distance function D) is smaller than some constant threshold MAXDIST. Thus, N_q define:

$$N_q = \{ q' \in N \mid D(q, q') \leq \text{MAXDIST} \} \tag{5}$$

Additionally, according to the algorithm, the authors try to connect q with all nodes in N_q in order to increase the distance from q and another configuration. They skip those nodes which are in the same connected component as q. By considering elements of N_q in this order they expect to maximize the chances of quickly connecting q to other configurations and, consequently, reduce the number of calls to the local planner, (Nice every successful connection results in merging two connected components into one). In (Kavraki et al., 1996), is found it a useful to bound the size of the set N_q by the constant K.

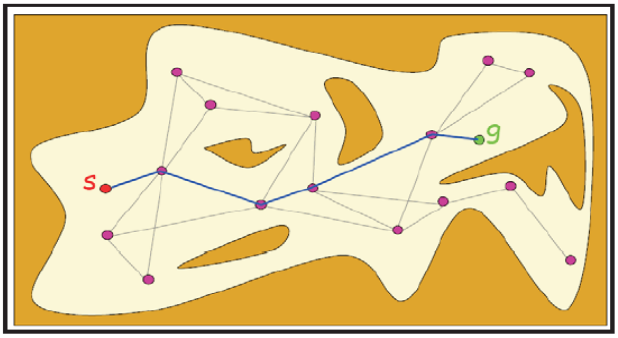


Fig. 5. The PRM is searched for a path from s to g through a graph which represent la connectivity of configuration space.

The distance function. The function D is used both to construct and sort the set N_q of candidate neighbours of each new node q. It should be defined so that, for any pair (q, q') of configurations, $D(q, q')$ reflects the possibility that the local planner will fail to compute a feasible path between these configurations. One possibility is to define $D(q, q')$ as a measure (area/volume) of the workspace region swept by the robot when it moves along the path computed by the local planner between q and q' in the absence of obstacles. Thus, each local planner would automatically induce its own specific distance function. Since exact computation of swept areas/volumes tends to be rather time-consuming, a rough but inexpensive measure of the swept-region gives better practical results. Very simple distance measure also seems to give good results. For example, when the general local planner described above is used to connect q and q', $D(q, q')$ may be defined as follows:

$$D(q, q') = max_{x \in robot} \| x(q) - x(q') \|, \tag{6}$$

Where x denotes a point on the robot, $x(q)$ is the position of x in the workspace when the robot is at configuration q, and $\| x(q) - x(q') \|$ is the Euclidean distance between $x(q)$ and $x(q')$.

3.2 The Obstacle Based PRM

In (Amato & Wu, 1996) presents a randomized roadmap method for motion planning for several DOF robots. The general approach follows traditional roadmap methods: during pre-processing a roadmap is built in C-space; planning consists of connecting the initial and goal configuration to the roadmap, and then finding a path in the roadmap between these two connection points. The main novelty in their approach is a method for generating roadmap candidate points. In particular, they attempt to generate candidate points uniformly distributed on the surface of each C-obstacle.

3.2.1 Sampling and connection strategies

This planner samples the obstacle surfaces without explicitly calculating those surfaces. The *OBPRM* generation strategy provides information which make possible to find a connection strategy where every node generated can be considered for connection with its k closest neighbours on each obstacle in the environment. *OBPRM* shows marked success in discovering and navigating narrow passages in C-space. They evaluate various sampling and connection strategies within the context of *OBPRM* in (Amato et al., 1998). A multi-strategy for connecting roadmap nodes where different local planners are used at different stages is shown to enhance the connectivity of the resulting roadmap significantly. The most common local planner used by *PRM* methods is a straight line in C-space. They evaluate distance metrics and local planner methods in (Amato et al., 1998) where they propose the $rotate - at - s$ local planner. $Rotate - at - s$ divides the usual straight line path into three straight line segments to be tested for collision. The first and final segments consist of pure translation while the intervening segment is pure rotation. The choice of strategy depends on the value of s ∈ [0, 1] which is in general provided by the user. Briefly, for each obstacle *X*:

PROTOTYOE NODE GENERATION

```
1. $C_{in}$:= collinding robot cfg with C-obtacle X
2. $D$:= $m$ random directions emanating out from $C_{in}$
3. for each $d \in D$
4. $C_{out}$:- free cfg in direction $d$ (if exists)
5. find contact cfg on ($C_{in}$, $C_{out}$) by binary search
6. end for
```

This example strategy was sufficient to establish the potential of the method. However, it is clear that more sophisticated node generation strategies are needed for more complex objects to produce a 'good' distribution of nodes in all the 'different' regions of C-free. Outline below are some of the methods we've implemented and tested. Keeping in the spirit of OBPRM, all these method employ information regarding the environment to guide node generation. Briefly, the methods are designed to generate three types of nodes: (i) *contact* configuration, (ii) *free* configuration (near contact surfaces), and (iii) *sets* of configuration (*shells*) *surrounding* C-obstacles.

Generating contact configurations. The node generation strategy used in the prototype version of OBPRM is attractive due to its simplicity and its efficiency (node generation typically accounted for 1-2% of pre-processing time). However, the distribution of the generated nodes is clear very sensitive to both the shape of the C-obstacle and to the *seed* (origin C_{in} for the binary search). That is, no single seed will yield a good distribution of configuration on the surface of the C-obstacle if its shape is not roughly spherical, and even if the C-obstacle is spherically shape, a seed configurations on the region of its surface closest to the seed.

GENERRATE CONTACT CONFIGURATION

```
1. $p_{rob}$:= point associated with root
2. $p_{ob\varepsilon}$:= point associated with obstacle of interest
3. $C_{in}$:= translate robot so $p_{rob}$ and $p_{ob\varepsilon}$ coincide an rotate robot randomly until
   collision
4. $d$:= random direction emanating out from $C_{in}$
5. $C_{out}$:= free cfg in direction $d$ (if exists)
6. Find contact on ($C_{in}$, $C_{out}$) by binary search
```

3.3 Visibility roadmap

In the visibility roadmap method proposed in (Laumond & Siméon, 2000), roadmap nodes are randomly generated just as in a *Basic − PRM* but connection is done as they are generated in the following way. For each generated node, if it can be connected to more than one currently existing connected component or to no currently existing connected component, it is retained and the connecting edges (if any) are added to the roadmap. If the node can be connected to only one of the existing connected component, it is discarded. Multiple connectable nodes are assumed to yield important information about the connectivity of free space. Isolated nodes are assumed to point to unexplored areas of free space. Singly connectable nodes are assumed to represent another sample in an already explored region and are discarded rather than allowing them to increase the size of the roadmap. Because nodes are generated randomly in the basic *PRM* manner, this method is not better than the basic *PRM* at handling narrow passages.

This method proposes a variant of the Probabilistic Roadmap (PRM) algorithm introduced in (Kavraki & Latombe, 1994) (and independently in (Overmars & Svestka, 1995) as the Probabilistic Path Planner). These algorithms generate collision-free configurations randomly and try to link them with a simple local path planning method. A roadmap is then generated, tending to capture the connectivity of the collision-free configuration space CS_{free}. This variant of these approaches takes advantage of the visibility notion. While usually each collision-free configuration generated by the PRM algorithm is integrated to the roadmap, our algorithm keeps only configurations which either connect two connected components of the roadmap, or are not "visible" by some so-called guard configurations. This approach computes roadmaps with small number of nodes. It integrates a termination condition related to the volume of the free space covered by the roadmap. Experimental comparison shows good performances in terms of computation time, especially when applied to configuration spaces with narrow passages.

Roadmaps A *roadmap* is a graph whose nodes are collision-free configurations.

Two nodes q and q' are adjacent if the path L (q,q') computed by the local method lies in CS_{free}. Roadmaps are used to solve motion planning problems by the so called *query* procedure: given two configurations q_{init} and $q_{goal,}$ the procedure first connects q_{init} (resp.q_{goal}) to the roadmap **R** if there exists q^*_{init} (resp. q^*_{goal}) goal such that L (q_{init}, q^*_{init}) ˝ CS_{free} and L (q_{goal}, q^*_{goal}) ˝ CS_{free} . Then the procedure searches for a path in the extended roadmap. If such a path exists, the solution of the motion planning problem appears as a path constituted by a finite connected sequence of subpaths computed by L .

Visibility domains For a given local method L , the visibility domain of a configuration q is defined as the domain:

$$V_{is}\,\mathsf{L}\,(q) = \{ q' \in CS_{free} \text{ such that } \mathsf{L}\,(q, q') \text{˝}\ CS_{free} \} \tag{7}$$

Configuration q is said to be the *guard* of V_{is} L (q).

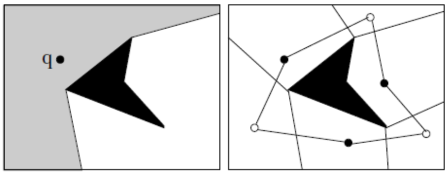


Fig. 6. Visibility domain of a configuration and the visibility roadmap defined by three guards nodes (black) and three connection nodes (white). Here paths L *(q, q')* are the straight-line segments [*q, q'*].

- **Free-space Coverage** A set of guards constitutes a *coverage* of CS_{free} if the union of their visibility domains covers the free space $CS_{free.}$ Note that the existence of finite coverage both depends on the shape of CS_{free} and on the local method L. Such finite coverage may not always exist. This issue is related to the notion of ϵ-*goodness* introduced in (Overmars & Svestka, 1995).

Visibility Roadmaps Consider now s visibility domains $V_{is} L\ (q_i)$ such that the *s* guards do not "see" mutually through the local method, i.e., $L\ (q_i, q'_j) \not\subset CS_{free}$ for any pair of guards (q_i, q'_j). Then we build the following graph **R**. Guards $\{ q_i \}_{i=1,s}$ are nodes of the graph. For any two intersecting visibility domains $V_{is} L\ (q_i)$ and $V_{is} L\ (q_j)$, we add a node *q*, called a *connection node*, and two edges (q, q'_i) and (q, q'_j). (see Figure 6) The graph **R** is said to be a *visibility roadmap.* **R** clearly verifies the following property:

Property: Let us assume a visibility roadmap R whose set of guards covers $CS_{free.}$ Let us consider any two configurations q_{init} and q_{goal} such as there exists a connected sequence of collision-free paths of type L between them. Then there is a guard node q_1 and a guard node q_2 in R such as: $q_{init} \in V_{is} L\ (q_1)$, $q_{goal} \in V_{is} L\ (q_2)$ with q_1 and q_2 lying in a same connected component of ***R****.*

The notion of visibility roadmap raises several comments:

- Since the definition of the visibility domains is related to a local method, it would have been better to use the term of "reachable domain". Both notions are identical when the local method simply computes straight line segments.
  We keep the word "visibility" because it is more intuitive.
- We consider implicitly that **R** is an undirected graph: that means that L is assumed to be symmetric.
- Finally the number of guards is not required to be optimal. Optimality refers to the well known and challenging art gallery problem (Goodman & O'Rourke, 1997)

Description The algorithm, called Visib-PRM iteratively processes two sets of nodes: *Guard* and *Connection*. The nodes of *Guard* belonging to a same connected component (i.e. connected by nodes of *Connection*) are gathered in subsets G_i.

Algorithm Visib-PRM

```
Guard ← Ø; Connection ← Ø; ntry ← 0
While (ntry < M)
    Select a random free configuration q
    $g_{vis}$ ← Ø; $G_{vis}$ ← Ø
    For all components $G_i$ of Guard do
        found ← FALSE
        For all nodes g of $G_i$ do
            If (q belongs to $V_{is}(g)$) then
                found ← TRUE
                If ($g_{vis}$ = Ø) then $g_{vis}$ ← g; $G_{vis}$ ← $G_i$
                Else /* q is a connection node */
                    Add q to Connection
                    Create edges (q, g) and (q, $g_{vis}$)
                    Merge components $G_{vis}$ and $G_i$
    until found =TRUE
    If ($g_{vis}$ = Ø)then /* q is a guard node */
        Add{q} to Guard; ntry ← 0
    Else ntry ← ntry +1
End
```

3.4 Rapidly Random Tee (RRT)

The idea behind this method is to incrementally construct a search tree that gradually improves the resolution but does not need to explicitly set any resolution parameters. A dense sequence of samples is used as a guide in the incremental construction of the tree. If this sequence is random, the resulting tree is called a rapidly exploring random tree (RRT). In general, this family of trees, whether the sequence is random or deterministic, will be referred to as rapidly exploring dense trees (RDTs) to indicate that a dense covering of the space is obtained.

This method uses the term *state space* to indicate a greater generality than is usually considered in path planning. For a standard problem, $X= C$, wich is the configuration space of a rigid body or system of bodies in a 2D or 3D world (Latombe, 1991).

For a given initial state, $x_{init,}$ an RRT, T, with K vertices is constructed as shown below:

```
1.  GENERATE_RRT ($x_{init}$, K, $\Delta t$)
2.  T.init($x_{init}$);
3.  For k=1 to K do
4.      $x_{rand}$ ← RANDOM_STATE();
5.      $x_{near}$ ← NEAREST_NEIGHBOR ($x_{rand}$, T);
6.      u ← SELECT_INPUT ($x_{rand}$, $x_{near}$);
```

```
7.     $x_{new} \leftarrow$ NEW_STATE ($x_{near, u}, \Delta t$)
8.     $T$.add_vertex($x_{new}$);
9.     $T$.add_edge($x_{nea}, x_{new, u}$);
10. Return $T$
```

3.4.1 Nice properties of RRTs
The key advantages of RRTs are: 1) the expansion of a RRT is heavily biased toward unexplored portion of the states space; 2) the distribution, leading to consistent behaviour; 3) an RRT probabilistically complete under very general condition; 4) the RRT algorithm is relatively simple, which facilitates performance analysis (this is also a preferred feature of probabilistic roadmaps): 5) an RRT always remains connected, even though the number of edges is minimal; 6) an RRT can be considered as a path planning module, which can be adapted and incorporated into a wide variety of planning system; 7) entire path planning algorithms can be constructed without requiring the ability to steer the system between two prescribed states, which greatly broadens the applicability of RRTs.

3.5 PRM based on Obstacles Geometry
The method is based on obstacles geometric for addressing narrow corridors problems (Antonio & Vallejo, 2004). The method takes advantage of geometric properties for computing free configurations in both, first approximation and improving phase.
A geometric representation of the workspace (the obstacles and the robot) is given through triangle meshes, including their positions and orientations. Each object in the environment is associated whit its *straightness* and *volume*. The following defines volume and straightness feature for each object in the environment.
"Straightness" indicates the direction of an object with respect its longest side. We represents this property using a vector denoted as v_i and its represents the direction of the straightness of each body. During the construction phase, we use this vector as the most convenient direction of the rotation axis of an object B_i. The direction of the rotation axis is very important, because, when the algorithm attempts to rotate the robot, it assumes that, the best selection to rotate the body is around v_i.
The "**volume"** of the body gives an approximation of the size of an object respect to the volume of its surrounding sphere. This feature is given as parameter for each object within the environment. The value is proportional to the surrounding sphere.

It is important to take into account that our algorithm takes advantage of the form of the bodies. Provided that an obstacle is built by smaller ones, "*straightness*" and "*volume*" are defined for each element part.

3.5.1 First approximation of the configuration space
The objective of the construction step is to obtain a reasonably connected graph and to make sure that most "*difficult*" regions in this space contain at least a few nodes. In particular, the objective of the first approximation of the roadmap is to obtain an initial sampling using an economic and fast process.

The nodes of R should constitute a uniform sampling of C_{free}. Every such configuration is obtained by drawing each of its coordinates from the interval of allowed values of the corresponding DOF using a uniform probability distribution over this interval. This sampling is computed using spheres which surround the objects. Figure 7 illustrate these sampling surrounding spheres of objects. During this stage, the center of gravity and the radius are used to compute the size of the surrounding sphere of each body in the environment. Two advantages can be seen: the collision detection algorithm, (which is a deterministic used to decide if the position and orientation where the robot is placed does not collide with any object into the environment), is reduced to verify intersection between spheres. And the fact that these free configurations is involved into a sphere, having the possibility to rotate to any direction.

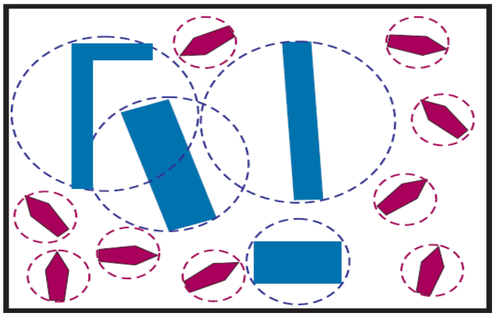


Fig. 7. The sampling of the roadmap is computed using a uniform distribution on the C-space and keeping the free configurations.

Collision detection is implemented using spheres, which means that we surround the robot and the obstacles within a sphere and the collision verification is reduced to compute the intersection among spheres. The computed configuration is labeled as *far configurations*). This property is used when the algorithm will tries to connect this configuration with its k-nearest neighbors.

3.5.2 Expanding the roadmap

If the number of nodes computed during the first approximation of the roadmap is large enough, the set N gives a fairly uniform covering of C_{free}. In easy scenes R is well connected. But in more constrained ones where C_{free} is actually connected, R often consists of a few large components and several small ones. It therefore does not effectively capture the connectivity of C_{free}. The purpose of the expansion is to add more nodes for facilitating the construction of the large components comprising as many nodes as possible for covering the most difficult narrow parts of C_{free}.

3.5.3 Elastic band algorithm

The "elastic band" algorithm attempts to find a free configuration from a collision one. To reach such goal, the algorithm moves the configuration using a small step for each iteration. This process is the result of applying the combination of both features (straightness and volume). First, the algorithm calculates the distance vector d_i between the obstacle position

and the configuration $c(B_i)$ position. Next, a value between the *MIN* and *MAX* parameters associated to the volume is computed and used as scalar quantity to increase or decrease the vector d_i. The following algorithm describes the operations used to compute the distance vector. Figure 8 presents a graphic illustration.
The main idea behind this scalar operation is to approach and move the robot away from the obstacle. To compute this operation, the process scales the d_i vector using the values computed with respect the *"volume"* feature to calculate the next position where the $c(B_i)$ will be placed. The following algorithm describes this process.

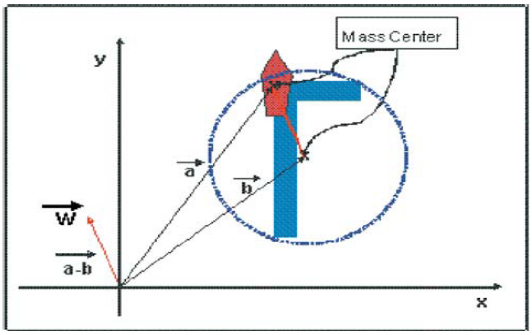


Fig. 8. Graphical representation of distance vector between the mass center of the Object and the mass center of the robot.

The elastic band process works with parallel and perpendicular configurations. Both types of configurations are computed around the obstacle. Configurations calculated in this phase are called *near configurations*. While the distance vector is computing the next configuration to be tested, the robot is rotated around its rotation axis, searching to find a free configuration and taking advantage of the *"straightness"*, sweeping the minor volume as result of this rotation.

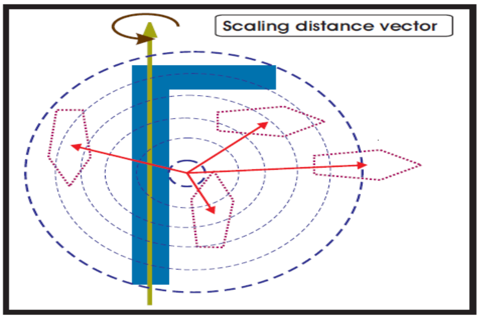


Fig. 9. Elastic Band approach and move the robot away from the obstacle to compute free configurations.

Figures 8 and Figure 9, shows how the parallel and perpendicular configurations are computed around the obstacle and how the scalar vector is changing, approaching and moving away the robot from the obstacle. The configurations marked with dots are calculated during the elastic band process.

Elastic Band Heuristic

```
1.    $B_i \leftarrow obstacle[i]$
2.    $q \leftarrow robot\_configuration$
3.    robot.get_parameters_obstacle ( MIN, MAX )
4.    $c_i\_init \leftarrow position\_of(B_i)$
5.    $k \leftarrow 0$
6.    $scalar \leftarrow MIN$
7.    $d_i \leftarrow distancebetween(c_i\ init, q)$
8.    do
9.        $scale(d_{i,,}\ scalar)$
10.       $q \leftarrow get\ configuration\ on(d_i)$
11.       $q \leftarrow rotate\ robot(q)$
12.       $k \leftarrow k + 1$
13.   while (q is in collision and $k \leq CTE$)
14.   if(q is free )
15.   $N \leftarrow N \cup q$
```

4. Collision Detection Algorithms

Collision detection is a fundamental problem in robotics, computer animation, physically-based modeling, molecular modeling and computer-simulated environments. In these applications, an object's motion is constrained by collisions with other objects and by other dynamic constraints. The problem has been well studied in the literature. A realistic simulation system, which couples geometric modeling and physical prototyping, can provide a useful toolset for applications in robotics, CAD/CAM design, molecular modeling, manufacturing design simulations, etc. In Figure 10. two sceneries are presented as a sample of environments that use collision detection. Such systems create electronic representations of mechanical parts, tools, and machines, which need to be tested for interconnectivity, functionality, and reliability. A fundamental component of such a system is to model object interactions precisely.

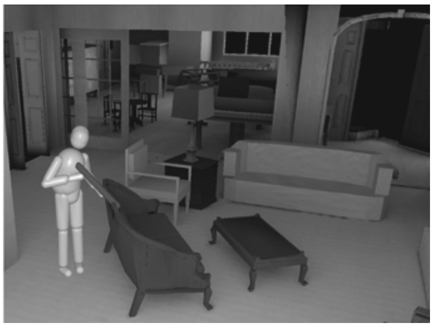


Fig. 10. 3-D Environments uses collision detection algorithms to restrict the movement ranks of several elements.

The interactions may involve objects in the simulation environment pushing, striking, or smashing other objects. Detecting collisions and determining contact points is a crucial step in portraying these interactions accurately. The most challenging problem in a simulation, namely the collision phase, can be separated into three parts: collision detection, contact area determination, and collision response.

4.1 Rapid version 2.01

RAPID is a robust and accurate polygon interference detection library for large environments composed of unstructured models (http://www.cs.unc.edu/~geom/OBB/OBBT.html).

- It is applicable to polygon soups - models which contain no adjacency information, and obey no topological constraints. The models may contain cracks, holes, self-intersections, and nongeneric (e.g. coplanar and collinear) configurations.
- It is numericaly robust - the algorithm is not subject to conditioning problems, and requires no special handling of nongeneric cases (such as parallel faces).

The RAPID library is free for non-commercial use. Please use this request form to download the latest version. It has a very simple user interface: the user need noncommercial use. Be familiar with only about five function calls. A C++ sample client program illustrates its use.
The fundamental data structure underlying RAPID is the OBBTree, which is a hierarchy of oriented bounding boxes (a 3D analog to the "strip trees" of Ballard). (Gottschalk et al., 1996).

5. GEMPA: Graphic Environment for Motion Planning Algorithms

Computer graphics has grown phenomenally in recent decades, progressing from simple 2-D graphics to complex, high-quality, three-dimensional environments. In entertainment, computer graphics is used extensively in movies and computer games. Animated movies are increasingly being made entirely with computers. Even no animated movies depend heavily on computer graphics to develop special effects. The capabilities of computer graphics in personal computers and home game consoles have now improved to the extent that low-cost systems are able to display millions of polygons per second.
The representation of different environments in such a system is used for a widely researched area, where many different types of problems are addressed, related to animation, interaction, and motion planning algorithms to name a few research topics. Although there are a variety of systems available with many different features, we are still a long way from a completely integrated system that is adaptable for many types of applications. This motivates us to create and build a visualization tool for planners capable of using physics-based models to generate realistic-looking motions. The main objective is to have a solid platform to create and develop algorithms for motion planning methods that can be launched into a digital environment. The developed of these tools allows to modify or to adapt the visualization tool for different kind of problems (Benitez & Mugarte, 2009).

5.1 GEMPA Architecture

GEMPA architecture is supported by necessary elements to represent objects, geometric transformation tools and visualization controls. These elements are integrated to reach initial goals of visualization and animation applied to motion planning problems.

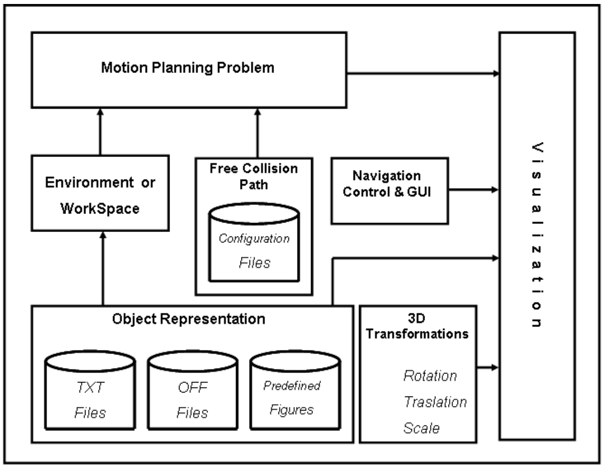


Fig. 11. Several modules are coupled to integrate the initial GEMPA architecture which offer interesting functionalities; visualization 3-D environments as well as animation of motion planning algorithms.

5.2 Recovering Objects Representation

People focus to solve problems using computer graphics, virtual reality and simulation of motion planning techniques used to recover information related to objects inside the environment through files which can storage information about triangle meshes. Hence, several objects can be placed on different positions and orientations to simulate a three-dimensional environment. There exist different formats to represent objects in three-dimensional spaces (3-D), however, two conventions used for many tools to represent triangle meshes are the most popular; objects based on *off - files* and objects based on *txt - files*. In motion planning community there exist benchmarks represented through this kind of files. GEMPA is able to load the triangle meshes used to represent objects from *txt* or *off – files*. On the other hand, GEMPA allows the user to built news environments using predefined figures as spheres, cones, cubes, etc. These figures are chosen from a option menu and the user build environments using translation, rotation and scale transformations. Each module on GEMPA architecture is presented in Figure 11 There, we can see that initially, the main goal is the visualization of 3-D environments and the animation of motion planning algorithms. In the case of visualization of 3-D environments, information is recovered form files and the user can navigate through the environment using mouse and keyboard controls. In the second case, the animation of motion planning algorithms, GEMPA needs information about the problem. This problem is described by two elements; the first one is called workspace, where obstacles (objects), robot representation and configuration (position and orientation) is recovered from files; the second, a set of free collision configuration conform a path, this will be used to animate the robot movement from initial to goal configuration. An example of 2D environment can be seen in Figure 12.

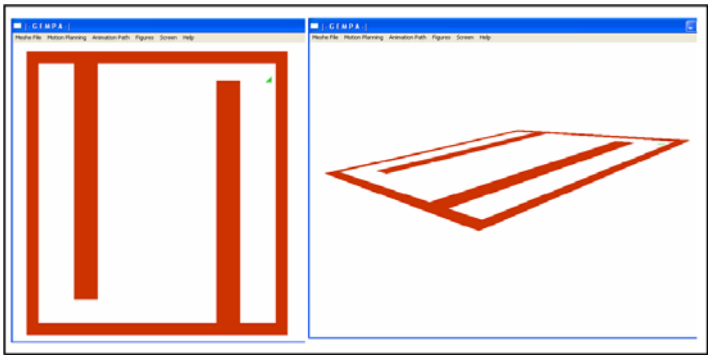

|-GEMPA-|
Meshe File Motion Planning Animation Path Figures Screen Help
|-GEMPA-|
Meshe File Motion Planning Animation Path Figures Screen Help

Fig. 12. Two different views of two-dimensional environment since the X-Y plane are painted.

5.3 GUI and Navigation Tools

GEMPA has incorporated two modes to paint an object; wire mode and solid mode. Next, Lambert illumination is implemented to produce more realism, and finally transparency effects are used to visualize the objects. Along the GUI, camera movements are added to facilitate the navigation inside the environment to display views from different locations. In Figure 13. Two illumination techniques are presented when GEMPA recover information since off-files to represent a human face.

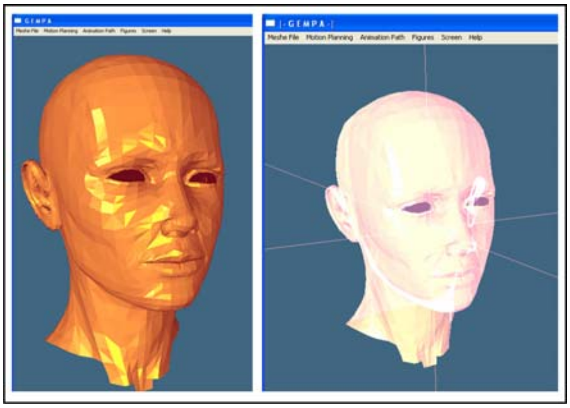

GEMPA
Meshe File Motion Planning Animation Path Figures Screen Help
|-GEMPA-|
Meshe File Motion Planning Animation Path Figures Screen Help

Fig. 13. Light transparency. In the left side, an object is painted using Lambert illumination, in the right side, transparency effect is applied on the object. Both features are used to give more realism the environment.

5.4 Simulation of Motion Planning Algorithms

Initially, only PRM for free flying objects are considered as an initial application of GEMPA. Taking into account this assumption, the workspace is conformed by a set of obstacles (objects) distributed on the environment, these objects has movement restrictions that mean that, the obstacles can not change their position inside the environment. In addition, an object that can move through the workspace is added to the environment and is called robot. The robot can move through the workspace using the free collision path to move from the initial configuration to the goal configuration. For PRM for free flying objects, only a robot can be defined and the workspace can include any obstacles as the problem need.
GEMPA also includes the capability to recover from an *environment – file* information about the position and orientation for each object inside a workspace including the robot configuration. Hence, GEMPA can draw each element to simulate the workspace associated. Therefore, initially GEMPA can recover information about the workspace, an example of this file can be see in Figure 14, where the environment file (left side), include initial and goal configuration for the robot, beside includes *x,y,z* parameter for position and (α, β, γ) parameters for orientation for every objects inside the workspace. Along with this *environment file*, a *configuration - file* can also be loaded to generate the corresponding animation of the free collision path. This *configuration - file* has the form presented in Figure 14 (right side). This file is conformed by *n* six-tuples (*x, y, z*,α, β, γ) to represent each configuration included in the free collision path.

```
Environment File

Robot
robot_angular.txt
0.0 0.0 0.0 0.0 0.0 0.0
1.0 20.0 0.0 0.7 1.2 0.8

Obstacle #1
bench_lamina_angosta_grande.txt
3.0 10 8.0 0.0 0.0 0.0

Obstacle #2
bench_lamina_angosta_grande.txt
3.0 10 -8.0 0.0 0.0 0.0
End
```

```
Configuration File
163
  5.0492   -0.7257   -3.2830   2.2233   6.1148   4.0886
  4.3394   -0.7601   -3.3466   2.1348   6.0969   4.1308
  3.6292   -0.7945   -3.4102   2.0463   6.0791   4.1730
  2.9189   -0.8289   -3.4737   1.9578   6.0613   4.2153
  2.2086   -0.8633   -3.5373   1.8692   6.0434   4.2575
  1.4983   -0.8977   -3.6008   1.7807   6.0256   4.2997
  0.7881   -0.9322   -3.6644   1.6922   6.0078   4.3420
  0.0778   -0.9666   -3.7280   1.6037   5.9900   4.3842
 -0.6324   -1.0010   -3.7915   1.5152   5.9721   4.4264
 -1.3427   -1.0354   -3.8551   1.4267   5.9543   4.4687
 -2.0529   -1.0698   -3.9186   1.3382   5.9365   4.5109
            .
            .
            .
            .
```

Fig. 14. On the left side, an example of environment - file (robot and obstacles representations) is presented, and on the right side a configuration file (free collision path) is shown.

Once GEMPA has recovered information about workspace and collision free path, the tool allows the user to display the animation on three different modes.

Mode 1: Animation painting all configurations.
Mode 2: Animation painting configurations using a step control.
Mode 3: Animation using automatic step.

From Figures 15 to Figure 18, we can see four different samples of motion planning problems which are considered as important cases. For each one, different views are

presented to show GEMPA's functionalities. Besides, we have presented motion planning problems with different levels of complexity.
In Figure 15. (Sample 1) The collision free path is painted as complete option and as animation option. In this sample a tetrahedron is considered as the robot.
Next, Figure 16: (Sample 2). A cube is presented as the robot for this motion planning problem. Here, GEMPA presents the flat and wire modes to paint the objects.
In Figure 17: (Sample 3). Presents a robot which has a more complex for and the problem becomes difficult to solve because the motion planning method needs to compute free configuration in the narrow corridor.
Finally in Figure 18: (Sample 4). Animation painting all configurations (left side), and animation using automatic step (right side) are displayed. Although the robot has not a more complex form, there are various narrow corridors inside the environment.

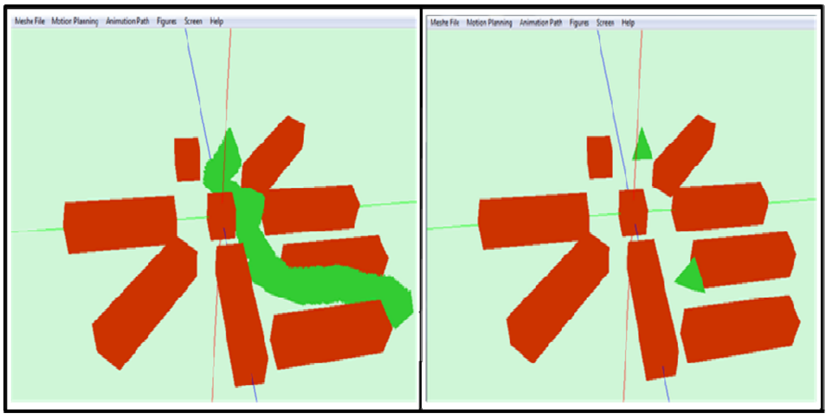

Meshe File Motion Planning Animation Path Figures Screen Help
Meshe File Motion Planning Animation Path Figures Screen Help

Fig. 15. Sample 1. The robot is presented as a tetrahedron.

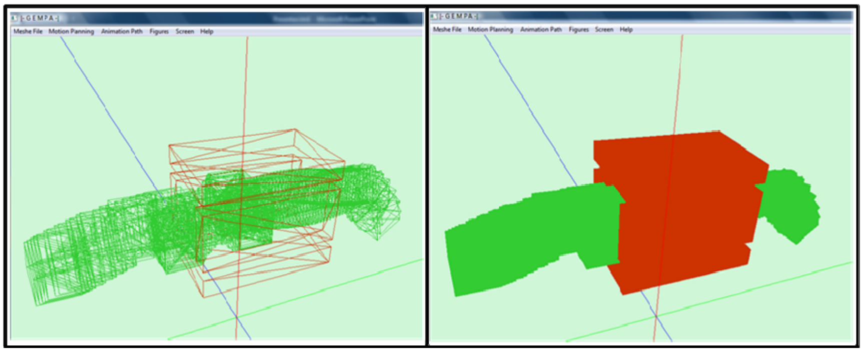

GEMPA
Meshe File Motion Panning Animation Path Figures Screen Help
GEMPA
Meshe File Motion Planning Animation Path Figures Screen Help

Fig. 16. Sample 2. The robot is presented as a cube.

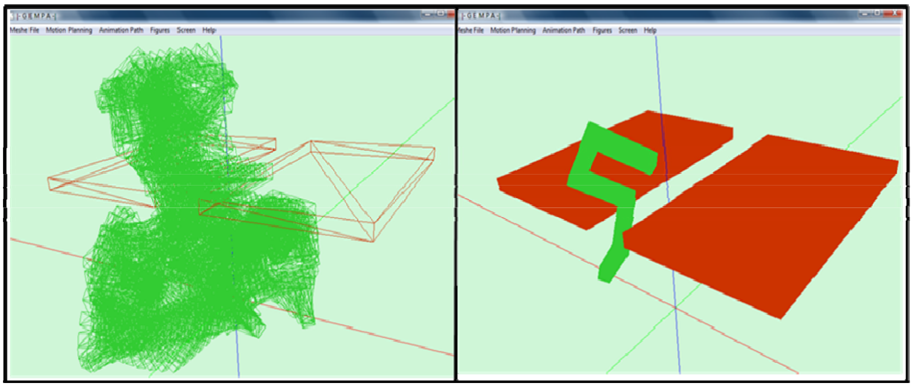

|-GEMPA-|
Meshe File Motion Planning Animation Path Figures Screen Help
|-GEMPA-|
Meshe File Motion Planning Animation Path Figures Screen Help

Fig. 17. Sample 3. The robot's form is more complex.

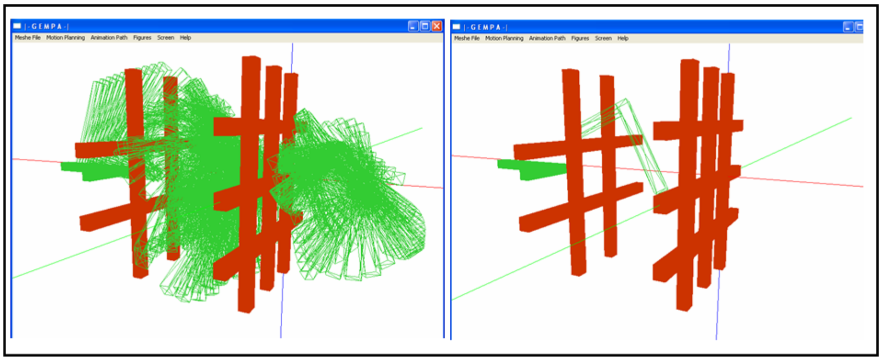

|-GEMPA-|
Meshe File Motion Planning Animation Path Figures Screen Help
|-GEMPA-|
Meshe File Motion Planning Animation Path Figures Screen Help

Fig. 18. Sample 4. More complex environment where various narrow corridors are presented.

6. References

Amato, N.; Bayazit, B. ; Dale, L.; Jones, C. &. Vallejo, D. (1998). Choosing good distance metrics and local planer for probabilistic roadmap methods. In in Procc.IEEE Int. Conf. Robot. Autom. (ICRA), pages 630–637.

Amato, N.; Bayazit, B. ; Dale, L.; Jones, C. &. Vallejo, D. (1998). Obprm: An obstaclebased prm for 3d workspaces. In in Procc. Int. Workshop on Algorithmic Fundation of Robotics (WAFR), pages 155–168.

Amato, N. M. & Wu, Y. (1996). A randomized roadmap method for path and manipulation planning. In In IEEE Int. Conf. Robot. and Autom., pages 113–120.

Amato, N. Motion ning puzzels benchmarks.

Benitez, A. & Mugarte, A. (2009). *GEMPA:Graphic Environment for Motion Planning Algorithm.* In Research in Computer Science, Advances in Computer Science and Engineering. Volumen 42.

Benitez, A. & Vallejo, D. (2004). *New Technique to Improve Probabilistic Roadmap Methods.* In proceedings of Mexican International Conference on Artificial Intelligence. (IBERAMIA) Puebla City, November 22-26, pag. 514-526.

Benitez, A.; Vallejo, D. & Medina, M.A. (2004). Prms based on obstacle's geometry.In In Proc. IEEE The 8th Conference on Intelligent Autonomous Systems, pages 592–599.

Boor, V.; Overmars, N. H. & van der Stappen, A. F. (1999). The gaussian sampling strategy for probabilistic roadmap planners. In In IEEE Int. Conf. Robot. And Autom., pages 1018–1023.

Chang, H. & Li, T. Y. (1995). Assembly maintainability study with motion planning. In In Proc. IEEE Int. Conf. on Rob. and Autom., pages 1012–1019.

Christoph, M. Hoffmann. Solid modeling. (1997). In Handbook of Discrete and ComputationalGeometry, pages 863,880. In Jacob E. Goodman and Joseph ORourke, editors Press, Boca Raton New York.

Goodman, J. & O'Rourke, J. (1997). Handbook of Discrete and Computational Geometry. CRC Press.In Computer Graphics (SIGGRAPH'94)., pages 395–408.

Kavraki, L. & Latombe, J.C. (1994). Randomized preprocessing of configuration space for path planning. In IEEE Int. Conf. Robot. and Autom, pages 2138–2145.

Kavraki, L. & Latombe, J.C. (1994). Randomized preprocessing of configuration space for fast path planning. In IEEE International Conference on Robotics and Automation, San Diego (USA), pp. 2138-2245.

Kavraki, L. E.; Svestka, P.; Latombe, J.-C. & Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. In IEEE Trans. Robot. & Autom, pages 566–580 .

Kavraki, L.; Kolountzakis, L. & Latombe, JC. (1996). Analysis of probabilistic roadmaps for path planning. In IEEE International Conference on Robotics and Automation, Minneapolis (USA), pp. 3020-3025.

Kavraki, L.E, J.C. Latombe, R. Motwani, & P. Raghavan. (1995). Randomized preprocessing of configuration space for path planning. In Proc. ACM Symp. on Theory of Computing., pages 353–362.

Koga, Y.; Kondo, K.; Kuffner, J. & Latombe, J.C. (1994). Planning motions with intentions.

Latombe, J.C. (1991). *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA.

Laumond, J. P. & Siméon, T. (2000). Notes on visibility roadmaps and path planning. In In Proc. Int. Workshop on Algorithmic Foundation of Robotics (WAFR), pages67–77.

M. LaValle and J. J. Kuffner. (1999). Randomized kinodynamic planning. In IEEE Int. Conf. Robot. and Autom. (ICRA), pages 473–479.

M. LaValle, J.H. Jakey, and L.E. Kavraki. (1999). A probabilistic roadmap approach for systems with closed kinematic chains. In IEEE Int. Conf. Robot. and Autom.

Overmars, M. & Svestka, P. (1995). A Probabilistic learning approach to motion Planning. In Algorithmic Foundations of Robotics of (WAFR94), K. Goldberg et al (Eds), pp. 19-37, AK Peters.

Overmars, M. & Svestka, P. (1994). A probabilistic learning approach to motion planning. In Proc. Workshop on Algorithmic Foundations of Robotics., pages 19–37.

Russell, S. & Norvig, P. (2003). Articial Intelligence: A Modern Approach. Pearson Education, Inc., Upper Saddle River, NJ.

Steven, M. LaValle. (2004). *Planning Algorithms*.

Tombropoulos, R.Z.; Adler, J.R. & Latombe, J.C. Carabeamer. (1999). A treatment planner for a robotic radiosurgical system with general kinematics. In Medical Image Analysis, pages 237–264.

Optimum Biped Trajectory Planning for Humanoid Robot Navigation in Unseen Environment

Hanafiah Yussof[1,2] and Masahiro Ohka[1]
[1]*Graduate School of Information Science, Nagoya University*
Japan
[2]*Faculty of Mechanical Engineering, Universiti Teknologi MARA*
Malaysia

1. Introduction

The study on biped locomotion in humanoid robots has gained great interest since the last decades (Hirai et. al. 1998, Hirukawa et. al., 2004, Ishiguro, 2007). This interest are motivated from the high level of mobility, and the high number of degrees of freedom allow this kind of mobile robot adapt and move upon very unstructured sloped terrain. Eventually, it is more desirable to have robots of human build instead of modifying environment for robots (Khatib et. al, 1999). Therefore, a suitable navigation system is necessary to guide the robot's locomotion during real-time operation. In fundamental robot navigation studies, robot system is normally provided with a map or a specific geometrical guidance to complete its tasks (Okada et al., 2003, Liu et al., 2002). However during operation in uncertain environment such as in emergency sites like an earthquake site, or even in a room that the robots never been there before, which is eventually become the first experience for them, robots needs some intelligence to recognize and estimate the position and structure of objects around them. The most important is robot must localize its position within this environment and decide suitable action based on the environment conditions. To archives its target tasks, the robot required a highly reliable sensory devices for vision, scanning, and touching to recognize surrounding. These problems have become the main concern in our research that deals with humanoid robot for application in built-for-human environment.
Operation in unseen environment or areas where visual information is very limited is a new challenge in robot navigation. So far there was no much achievement to solve robot navigation in such environments. In previous research, we have proposed a contact interaction-based navigation strategy in a biped humanoid robot to operate in unseen environment (Hanafiah et al., 2008). In this chapter, we present analysis results of optimum biped trajectory planning for humanoid robot navigation to minimize possibility of collision during operation in unseen environment. In this analysis, we utilized 21-dof biped humanoid robot Bonten-Maru II. Our aim is to develop reliable walking locomotion in order

to support the main tasks in the humanoid robot navigation system. Fig. 1 shows diagram of humanoid robot Bonten-Maru II and its configurations of dofs.

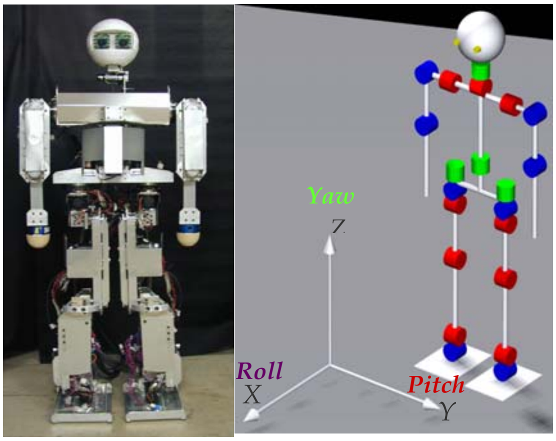


Fig. 1. Humanoid Robot Bonten-Maru II and its configuration of dofs.

It is inevitable that stable walking gait strategy is required to provide efficient and reliable locomotion for biped robots. In the biped locomotion towards application in unseen environment, we identified three basic motions: walk forward and backward directions, side-step to left and right, and yawing movement to change robot's orientation. In this chapter, at first we analyzed the joint trajectory generation in humanoid robot legs to define efficient gait pattern. We present kinematical solutions and optimum gait trajectory patterns for humanoid robot legs. Next, we performed analysis to define efficient walking gait locomotion by improvement of walking speed and travel distance without reducing reduction-ratio at joint-motor system. This is because sufficient reduction-ratio is required by the motor systems to supply high torque to the robot's manipulator during performing tasks such as object manipulation and obstacle avoidance. We also present optimum yawing motion strategy for humanoid robot to change its orientation within confined space. The analysis results were verified with simulation and real-time experiment with humanoid robot Bonten-Maru II.
Eventually, to safely and effectively navigate robots in unseen environment, the navigation system must feature reliable collision checking method to avoid collisions. In this chapter, we present analyses of collision checking using the robot arms to perform searching, touching and grasping motions in order to recognize its surrounding condition. The collision checking is performed in searching motion of the robot's arms that created a radius of detection area within the arm's reach. Based on the searching area coverage of the robot arms, we geometrically analyze the robot biped motions using Rapid-2D CAD software to identify the ideal collision free area. The collision free area is used to calculate maximum biped step-length when no object is detected. Consequently the robot control system created an absolute collision free area for the robot to generate optimum biped trajectories. In case of object is detected during searching motion, the robot arm will touch and grasp the object surface to define self-localization, and consequently optimum step-length is refined.

Verification experiments were conducted using humanoid robot Bonten-Maru II to operate in a room with walls and obstacles was conducted. In this experiment, the robot visual sensors are not connected to the system. Therefore the robot locomotion can only rely on contact interaction of the arms that are equipped with force sensors.

2. Short Survey on Humanoid Robot Navigation

Operation in unseen environment or areas where visual information is very limited is a new challenge in robot navigation. So far there was no much achievement to solve robot navigation in such environments. In normal conditions, it is obvious that a navigation system that applies non-contact sensors such as vision sensors provides intensive information about the environment (Sagues & Guerrero, 1999). However, robots cannot just rely on this type of sensing information to effectively work and cooperate with humans. For instance, in real applications the robots are likely to be required to operate in areas where vision information is very limited, such as in a dark room or during a rescue mission at an earthquake site (Diaz et. al., 2001). Moreover vision sensors have significant measurement accuracy problems resulting from technical problems such as low camera resolution and the dependence of stereo algorithms on specific image characteristics. Furthermore, the cameras are normally located at considerable distance from objects in the environment where operation takes place, resulting in approximate information of the environment.
In addition to the above, a laser range finder has also been applied in a robot navigation system (Thompson et. al., 2006). This sensor is capable of producing precise distance information and provides more accurate measurements compared with the vision sensor. However, it is impractical to embed this type of sensor with its vision analysis system in a walking robot system because of its size and weight (Okada et. al., 2003). A navigation system that applies contact-based sensors is capable of solving the above problems, particularly for a biped walking robot system (Hanafiah et. al., 2007). This type of sensor can accurately gauge the structure of the environment, thus making it suitable to support current navigation systems that utilize non-contact sensors. Furthermore, the system architecture is simpler and can easily be mounted on the walking robot body.
Eventually, to safely and effectively navigate robots in unseen environment, the navigation system must feature reliable collision checking method to avoid collisions. To date, in collision checking and prediction research, several methods such as vision based local floor map (Okada et al., 2003, Liu et al., 2002) and cylinder model (Guttmann et al., 2005) have been proposed for efficient collision checking and obstacle recognition in biped walking robot. In addition, Kuffner (Kuffner et al., 2002) have used fast distance determination method for self-collision detection and prevention for humanoid robots. This method is for convex polyhedra in order to conservatively guarantee that the given trajectory is free of self-collision. However, to effectively detect objects based on contact-based sensors, such methods are not suitable because they are mostly based on assumption of environment conditions acquired by non-contact sensors such as vision and laser range sensors.
Several achievements have been reported related with navigation in humanoid robots. Ogata have proposed human-robot collaboration based on quasi-symbolic expressions applying humanoid on static platform named Robovie (Ogata et al., 2005). This work combined non-contact and contact sensing approach in collaboration of human and robot during navigation tasks. This is the closest work with the approach used in this research.

However Ogata use humanoid robot without leg. On the other hand, related with biped humanoid robot navigation, the most interesting work was presented by Stasse where visual 3D Simultaneous Localization and Mapping (SLAM) was used to navigate HRP-2 humanoid robot performing visual loop-closing motion (Stasse et al., 2006). In other achievements, Gutmann (Gutmann et al., 2005) have proposed real-time path planning for humanoid robot navigation. The work was evaluated using QRIO Sony's small humanoid robot equipped with stereo camera. Meanwhile, Seara have evaluated methodological aspects of a scheme for visually guided humanoid robot navigation using simulation (Seara et al., 2004). Next, Okada have proposed humanoid robot navigation system using vision based local floor map (Okada et al., 2003). Related with sensory-based biped walking, Ogura (Ogura et al., 2004) has proposed a sensory-based biped walking motion instruction strategy for humanoid robot using visual and auditory sensors to generate walking patterns according to human orders and to memorize various complete walking patterns. In previous research, we have proposed a contact interaction-based navigation strategy in a biped humanoid robot to operate in unseen environment (Hanafiah et. al., 2008). In this chapter, we present analysis results of optimum biped trajectory planning for humanoid robot navigation to minimize possibility of collision during operation in unseen environment.

3. Simplification of Kinematics Solutions

A reliable trajectory generation formulations will directly influence stabilization of robot motion especially during operation in unseen environment where the possibility of unstable biped walking due to ground condition and collision with unidentified objects are rather high if compared to operation in normal condition. In this chapter, at first we analyzed the joint trajectory generation in humanoid robot legs to define efficient gait pattern. We present kinematical solutions and optimum gait trajectory patterns for humanoid robot legs.

Eventually, formulations to generate optimum trajectory in articulated joints and manipulators are inevitable in any types of robots, especially for legged robot. Indeed, the most sophisticated forms of legged motion are that of biped gait locomotion. However calculation to solve kinematics problems to generate trajectory for robotic joints is a complicated and time-consuming study, especially when it involves a complex joint structure. Furthermore, computation of joint variables is also needed to compute the required joint torques for the actuators. In current research, to generate optimum robot trajectory, we simplified kinematics formulation to generate trajectory for each robot joint in order to reduce calculation time and increase reliability of robot arms and legs motions. This is necessary because during operation in unseen environment, robot will mainly rely on contact interaction using its arms. Consequently, an accurate and fast respond of robot's both legs are very important to maintain stability of its locomotion.

We implemented a simplified approach to solving inverse kinematics problems by classifying the robot's joints into several groups of joint coordinate frames at the robot's manipulator. To describe translation and rotational relationship between adjacent joint links, we employ a matrix method proposed by Denavit-Hartenberg (Denavit & Hartenberg, 1955), which systematically establishes a coordinate system for each link of an articulated chain. Since this chapter focusing on biped trajectory, we present kinematical analysis of 6-dofs leg in the humanoid robot Bonten-Maru II body.

3.1 Kinematical Solutions of 6-DOFs Leg

Each of the legs has six dofs: three dofs (yaw, roll and pitch) at the hip joint, one dof (pitch) at the knee joint and two dofs (pitch and roll) at the ankle joint. In this research, we solve only inverse kinematics calculations for the robot leg. Figure 2 shows the structure and configuration of joints and links in the robot's leg. A reference coordinate is taken at the intersection point of the 3-dofs hip joint.

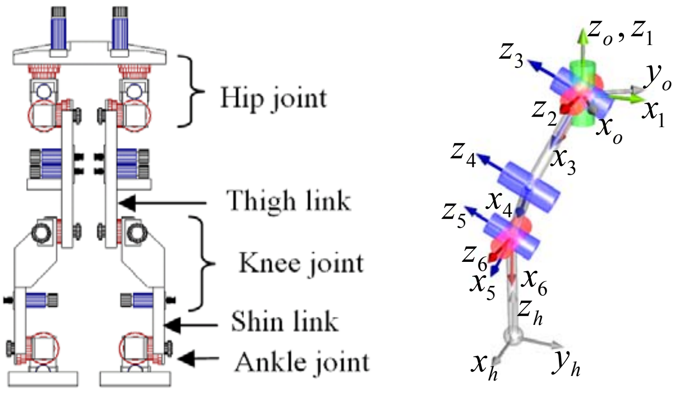


Fig. 2. Leg structure of Bonten-Maru II and configurations of joint coordinates.

Link	$\theta_{i\text{leg}}$	d	α	l
0	$\theta_{1\text{leg}}+90^o$	0	0	0
1	$\theta_{2\text{leg}}-90^o$	0	90^o	0
2	$\theta_{3\text{leg}}$	0	90^o	0
3	$\theta_{4\text{leg}}$	0	0	l_1
4	$\theta_{5\text{leg}}$	0	0	l_2
5	$\theta_{6\text{leg}}$	0	-90^o	0
6	0	0	0	l_3

Table 1. Link parameters of the 6-dofs humanoid robot leg.

In solving calculations of inverse kinematics for the leg, the joint coordinates are divided into eight separate coordinate frames as listed bellow:

Σ_0 : Reference coordinate.
Σ_1 : Hip yaw coordinate.
Σ_2 : Hip roll coordinate.
Σ_3 : Hip pitch coordinate.
Σ_4 : Knee pitch coordinate.
Σ_5 : Ankle pitch coordinate.
Σ_6 : Ankle roll coordinate.
Σ_h : Foot bottom-center coordinate.

Figure 2 also shows a model of the robot leg that indicates the configurations and orientation of each set of joint coordinates. Here, link length for the thigh is l_1, while for the shin it is l_2. Link parameters for the leg are defined in Table 1. From the Denavit-Hartenberg convention mentioned above, definitions of the homogeneous transform matrix of the link parameters can be described as follows:

$$ {}^{0}_{h}\mathbf{T} = \mathrm{Rot}(z_i,\theta_i)\mathrm{Trans}(0,0,d_i)\mathrm{Trans}(l_i,0,0)\mathrm{Rot}(x_i,\alpha_i). \tag{1}$$

Here, variable factor θ_i is the joint angle between the x_{i-1} and the x_i-axes measured about the z_i axis; d_i is the distance from the x_{i-1} axis to the x_i axis measured along the z_i axis; α_i is the angle between the z_i axis to the z_{i-1} axis measured about the x_{i-1} axis, and l_i is the distance from the z_i axis to the z_{i-1} axis measured along the x_{i-1} axis. Referring to Fig. 2, the transformation matrix at the bottom of the foot (${}^{6}_{h}\mathbf{T}$) is an independent link parameter because the coordinate direction is changeable. Here, to simplify the calculations, the ankle joint is positioned so that the bottom of the foot settles on the floor surface. The leg's orientation is fixed from the reference coordinate so that the third row of the rotation matrix at the leg's end becomes like equation (2).

$$P_{zleg} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \tag{2}$$

Furthermore, the leg's links are classified into three groups to short-cut the calculations, where each group of links is calculated separately as follows:
i) From link 0 to link 1 (Reference coordinate to coordinate joint number 1).
ii) From link 1 to link 4 (Coordinate joint no. 2 to coordinate joint no. 4).
iii) From link 4 to link 6 (Coordinate joint no. 5 to coordinate at the bottom of the foot).
Basically, i) is to control leg rotation at the z-axis, ii) is to define the leg position, while iii) is to decide the leg's end-point orientation. A coordinate transformation matrix can be arranged as following.

$$ {}^{0}_{h}\mathbf{T} = {}^{0}_{1}\mathbf{T}\,{}^{1}_{4}\mathbf{T}\,{}^{4}_{h}\mathbf{T} = ({}^{0}_{h}\mathbf{T})({}^{1}_{2}\mathbf{T}\,{}^{2}_{3}\mathbf{T}\,{}^{3}_{4}\mathbf{T})({}^{4}_{5}\mathbf{T}\,{}^{5}_{6}\mathbf{T}\,{}^{6}_{h}\mathbf{T}) \tag{3}$$

Here, the coordinate transformation matrices for ${}^{1}_{4}\mathbf{T}$ and ${}^{4}_{h}\mathbf{T}$ can be defined as (4) and (5), respectively.

$$ {}^{1}_{4}\mathbf{T} = {}^{1}_{2}\mathbf{T}\,{}^{2}_{3}\mathbf{T}\,{}^{3}_{4}\mathbf{T} $$

$$ = \begin{bmatrix} s_2c_{34} & -s_2s_{34} & -c_2 & l_1s_2c_3 \\ -s_{34} & -c_{34} & 0 & -l_1s_3 \\ -c_2c_{34} & c_2s_{34} & -s_2 & -l_1c_2c_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$ {}^{4}_{h}\mathbf{T} = {}^{4}_{5}\mathbf{T}\,{}^{5}_{6}\mathbf{T}\,{}^{6}_{h}\mathbf{T} $$

$$ = \begin{bmatrix} c_5c_6 & -c_5s_6 & -s_5 & l_2 + l_3c_5c_6 \\ s_5c_6 & -s_5s_6 & c_5 & l_3s_5c_6 \\ -s_6 & -c_6 & 0 & -l_3s_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

The coordinate transformation matrix for ${}^{0}_{h}\mathbf{T}$, which describes the leg's end-point position and orientation, can be shown with the following equation.

$$ {}^{0}_{h}\mathbf{T} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6} $$

From equation (2), the following conditions were satisfied.

$$ r_{13} = r_{23} = r_{31} = r_{32} = 0 \ , \ r_{33} = 1 \tag{7} $$

Hence, joint rotation angles $\theta_{1\text{leg}}$~$\theta_{6\text{leg}}$ can be defined by applying the above conditions. First, considering i), in order to provide rotation at the z-axis, only the hip joint needs to rotate in the yaw direction, specifically by defining $\theta_{1\text{leg}}$. As mentioned earlier, the bottom of the foot settles on the floor surface; therefore, the rotation matrix for the leg's end-point measured from the reference coordinate can be defined by the following equation.

$$ {}^{0}_{h}\mathbf{R} = \mathrm{Rot}(z, \theta_{1\text{leg}}) $$

$$ = \begin{bmatrix} c\theta_{1\text{leg}} & -s\theta_{1\text{leg}} & 0 \\ s\theta_{1\text{leg}} & c\theta_{1\text{leg}} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & 0 \\ r_{21} & r_{22} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8} $$

Here, $\theta_{1\text{leg}}$ can be defined as below.

$$ \theta_{1\text{leg}} = \mathrm{atan2}(r_{21}, r_{22}) \tag{9} $$

Next, considering ii), from the obtained result of $\theta_{1\text{leg}}$, ${}^{0}_{h}\mathbf{T}$ is defined in (9).

$$ {}^{0}_{h}\mathbf{T} = \begin{bmatrix} -s_1 & -c_1 & 0 & P_{x\text{leg}} \\ c_1 & -s_1 & 0 & P_{y\text{leg}} \\ 0 & 0 & 1 & P_{z\text{leg}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10} $$

Here, from constrain orientation of the leg's end point, the position vector of joint 5 is defined as follows in (11), and its relative connection with the matrix is defined in (12). Next, equation (13) is defined relatively.

$$ {}^{0}\mathbf{P}_5 = {}^{0}_{4}\mathbf{T}\,{}^{4}\mathbf{P}_5 = \begin{bmatrix} P_{x\text{leg}} & P_{y\text{leg}} & P_{z\text{leg}} - l_3 \end{bmatrix}^T , \tag{11} $$

$$ {}^{1}_{4}T\,{}^{4}\hat{P}_5 = {}^{0}_{1}T^{-1}\,{}^{0}\hat{P}_5 \tag{12} $$

$$ \begin{bmatrix} s_2c_{34} & -s_2s_{34} & -c_2 & l_1s_2c_3 \\ -s_{34} & -c_{34} & 0 & -l_1s_3 \\ -c_2c_{34} & c_2s_{34} & -s_2 & -l_1c_2c_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} l_2 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -s_1 & c_1 & 0 & 0 \\ -c_1 & -s_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z - l_3 \\ 1 \end{bmatrix} \tag{13} $$

Therefore,

$$ \begin{bmatrix} \hat{P}_{x\text{leg}} \\ \hat{P}_{y\text{leg}} \\ \hat{P}_{z\text{leg}} \end{bmatrix} = \begin{bmatrix} s_2(l_1c_3 + l_2c_{34}) \\ -(l_1c_3 + l_2s_{34}) \\ -c_2(l_1c_3 + l_2c_{34}) \end{bmatrix} . \tag{14} $$

To define joint angles $\theta_{2\text{leg}}$, $\theta_{3\text{leg}}$, $\theta_{4\text{leg}}$, equation (14) is used. Therefore, the rotation angles are defined as the following equations:

$$\theta_{4\text{leg}} = \text{atan2}\left(\pm\sqrt{1-C^2}, C\right) \tag{15}$$

$$\theta_{3\text{leg}} = \text{atan2}\left(\hat{p}_{xz\text{leg}}, \hat{p}_{y\text{leg}}\right) + \text{atan2}(k_1, k_2) \tag{16}$$

$$\theta_{2\text{leg}} = \text{atan2}\left(\hat{p}_{x\text{leg}}, \hat{p}_{z\text{leg}}\right). \tag{17}$$

Eventually, $C, \hat{p}_{xz\text{leg}}, k_1, k_2$ are defined as follows:

$$C = \frac{\hat{p}_{x\text{leg}}^{\;2} + \hat{p}_{y\text{leg}}^{\;2} + \hat{p}_{z\text{leg}}^{\;2} - (l_1^2 + l_2^2)}{2l_1l_2} \tag{18}$$

$$\hat{p}_{xz\text{leg}} = \sqrt{\hat{p}_{x\text{leg}}^{\;2} + \hat{p}_{z\text{leg}}^{\;2}} \tag{19}$$

$$k_1 = l_1 + l_2c_4,\;\; k_2 = -l_2s_4 \tag{20}$$

Finally, considering iii), joint angles $\theta_{5\text{leg}}$ and $\theta_{6\text{ leg}}$ are defined geometrically by the following equations:

$$\theta_{5\text{leg}} = -\theta_{3\text{leg}} - \theta_{4\text{leg}} \tag{21}$$

$$\theta_{6\text{leg}} = -\theta_{2\text{leg}}\,. \tag{22}$$

3.2 Interpolation and Gait Trajectory Pattern

A common way of making a robot's manipulator to move from start point to end point in a smooth, controlled fashion is to have each joint to move as specified by a smooth function of time *t*. Each joint starts and ends its motion at the same time, thus the robot's motion appears to be coordinated. In this research, we employ degree-5 polynomial equations to solve interpolation from start point P_0 to end point P_f. Degree-5 polynomial equations provides smoother gait trajectory compared to degree-3 polynomial equations which commonly used in robotic control. Velocity and acceleration at P_0 and P_f are defined as zero; only the position factor is considered as a coefficient for performing interpolation.

$$P(t) = a_0 + a_1t + a_2t^2 + a_3t^3 + a_4t^4 + a_5t^5 \tag{23}$$

Time factor at P_0 and P_f are describe as $t_0 = 0$ and t_f, respectively. Here, boundary condition for each position, velocity and acceleration at P_0 and P_f are shown at following equations.

$$\left.\begin{aligned}
&P(0) = a_0 = P_o \\
&\dot{P}(0) = a_1 = \dot{P}_o \\
&\dot{P}(0) = 2a_2 = \ddot{P}_o \\
&P(t_f) = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 + a_4 t_f^4 + a_5 t_f^5 = P_f \\
&\dot{P}(t_f) = a_1 + 2a_2 t_f + 3a_3 t_f^2 + 4a_4 t_f^3 + 5a_5 t_f^4 = \dot{P}_f \\
&\ddot{P}(t_f) = 2a_2 + 6a_3 t_f + 12a_4 t_f^2 + 20a_5 t_f^3 = \ddot{P}_f
\end{aligned}\right\} \tag{24}$$

Here, coefficient a_i (i = 0,1,2,3,4,5) are defined by solving deviations of above equations. Results of the deviations are shown at below equations.

$$\left.\begin{aligned}
&a_0 = y_o \\
&a_1 = \dot{y}_o \\
&a_2 = \frac{1}{2}\ddot{y}_o \\
&a_3 = \frac{1}{2t_f^3}\{20(y_f - y_0) - (8\dot{y}_f + 12\dot{y}_o)t_f + (\ddot{y}_f - 3\ddot{y}_o)t_f^2\} \\
&a_4 = \frac{1}{2t_f^4}\{-30(y_f - y_o) + (14\dot{y}_f + 16\dot{y}_o)t_f - (2\ddot{y}_f - 3\ddot{y}_o)t_f^2\} \\
&a_5 = \frac{1}{2t_f^5}\{12(y_f - y_o) - 6(\dot{y}_f + \dot{y}_o)t_f + (\ddot{y}_f - \ddot{y}_o)t_f^2\}
\end{aligned}\right\} \tag{25}$$

As mentioned before, velocity and acceleration at P_0 and P_f were considered as zero, as shown in (26).

$$\dot{P}(0) = \ddot{P}(0) = \dot{P}(t_f) = \ddot{P}(t_f) = 0\,. \tag{26}$$

Generation of motion trajectories from points P_0 to P_f only considered the position factor. Therefore, by given only positions data at P_0 and P_f, respectively described as y_0 and y_f, coefficients a_i (i = 0,1,2,3,4,5) were solved as below.

$$\left.\begin{aligned}
&a_0 = y_o \\
&a_1 = 0 \\
&a_2 = 0 \\
&a_3 = \frac{10}{t_f^3}(y_f - y_o) \\
&a_4 = -\frac{15}{t_f^4}(y_f - y_o) \\
&a_5 = \frac{6}{t_f^5}(y_f - y_o)
\end{aligned}\right\} \tag{27}$$

Finally, degree-5 polynomial function is defined as following equation.

$$y(t) = y_o + 10(y_f - y_o)u^3 - 15(y_f - y_o)u^4 + 6(y_f - y_o)u^5 \tag{28}$$

Where, $$u = \frac{t}{t_f} = \frac{current\ time}{motion\ time}. \tag{29}$$

These formulations provide smooth and controlled motion trajectory to the robot's manipulators during performing tasks in the proposed navigation system. Consequently, to perform a smooth and reliable gait, it is necessary to define step-length and foot-height during transferring one leg in one step walk. The step-length is a parameter value that can be adjusted and fixed in the control system. On the other hand, the foot-height is defined by applying ellipse formulation, like shown in gait trajectory pattern at Fig. 3. In case of walking forward and backward, the foot height at *z*-axis is defined in (30). Meanwhile during side steps, the foot height is defined in (31).

$$z = b\left(1 - \frac{x^2}{a^2}\right)^{\frac{1}{2}} - h \tag{30}$$

$$z = b\left(1 - \frac{y^2}{a^2}\right)^{\frac{1}{2}} - h \tag{31}$$

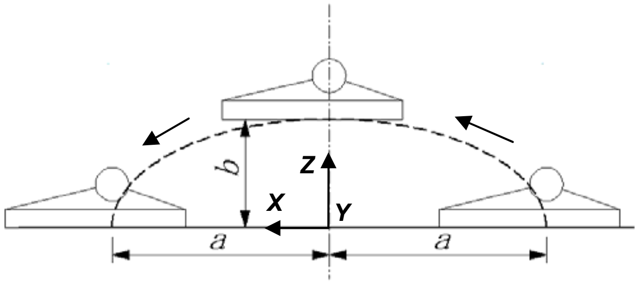


Fig. 3. Gait trajectory pattern of robot leg.

Here, *h* is hip-joint height from the ground. In real-time operation, biped locomotion is performed by giving the leg's end point position to the robot control system so that joint angle at each joint can be calculated by inverse kinematics formulations. Consequently the joint rotation speed and biped trajectory pattern are controlled by formulations of interpolation. By applying these formulations, each gait motion is performed in smooth and controlled trajectory.

4. Analysis of Biped Trajectory Locomotion

It is inevitable that stable walking gait strategy is required to provide efficient and reliable locomotion for biped robots. In the biped locomotion towards application in unseen environment, we identified three basic motions: walk forward and backward directions, side-step to left and right, and yawing movement to change robot's orientation. In this section, we performed analysis to define efficient walking gait locomotion by improvement of walking speed and travel distance without reducing reduction-ratio at joint-motor system.

This is because sufficient reduction-ratio is required by the motor systems to supply high torque to the robot's manipulator during performing tasks such as object exploration and obstacle avoidance. We also present optimum yawing motion strategy for humanoid robot to change its orientation within confined space.

4.1 Human Inspired Biped Walking Characteristics

Human locomotion stands out among other forms of biped locomotion chiefly in terms of the dynamic systems point of view. This is due to the fact that during a significant part of the human walking motion, the moving body is not in static equilibrium. The ability for humans to perform biped locomotion is greatly influenced by their learning ability (Dillmann, 2004, Salter et al., 2006). Apparently humans cannot walk when they are born but they can walk without thinking that they are walking as years pass by. However, robots are not good at learning. They are what they are programmed to do. In order to perform biped locomotion in robots, we must at first understand human's walking pattern and then develop theoretical strategy to perform the correct joint trajectories synthesis on the articulated chained manipulators at the robot's legs.
Figure 4 shows divisions of the gait cycle in human which focusing on right leg. Each gait cycle is divided into two periods, stance and swing. These often are called gait phase. Stance is the term used to designate the entire period during which the foot is on the ground. Both start and end of stance involve a period of bilateral foot contact with the floor (double stance), while the middle portion of stance has one foot contact. Stance begins with initial contact of heel strike, also known as initial double stance which begins the gait circle. It is the time both feet are on the floor after initial contact. The word swing applies to the time the foot is in the air for limb advancement. Swing begins as the foot is lifted from the floor. It was reported that the gross normal distribution of the floor contact periods is 60% for stance and 40% for swing (Perry, 1992). However, the precise duration of these gait cycle intervals varies with the person's walking velocity. The duration of both gait periods (stance and swing) shows an inverse relationship to walking speed. That is, both total stance and swing times are shortened as gait velocity increases. The change in stance and swing times becomes progressively greater as speed slows.

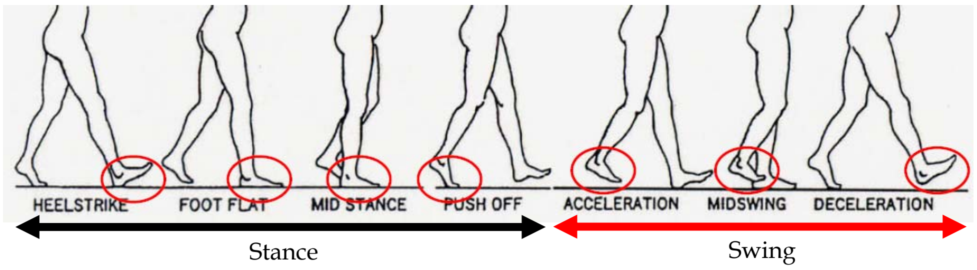


Fig. 4. Walking gait cycle in human.

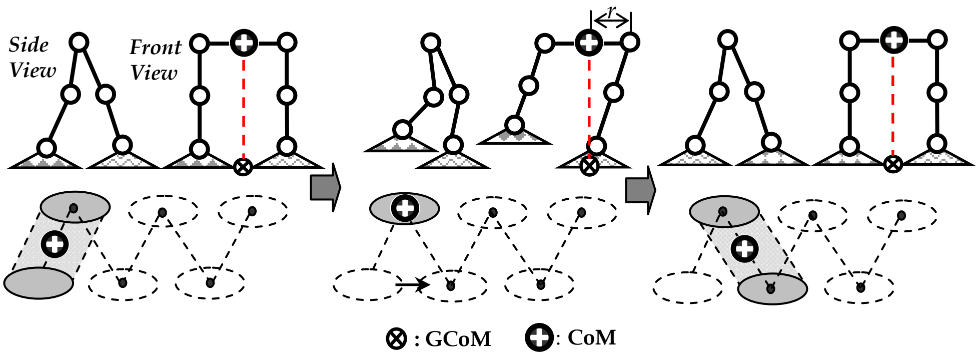


Fig. 5. Static walking model for biped robot in one cycle.

In contrast, for a biped robot two different situations arise in sequence during the walking motion: the statically stable double-support phase in which the whole structure of the robot is supported on both feet simultaneously, and the statically unstable single-support phase when only one foot is in contact with the ground, while the other foot is being transferred from back to front. Biped walking robot can be classified by its gait. There are two major research areas in biped walking robot: the static gait and dynamic gait. To describe gait motion in walking robots, it is easier to at first look at the static walking pattern point of view. In static walking pattern, two terms are normally used: Center of Mass (CoM) and Ground Projection of Center of Mass (GCoM). It is understood that to realize a stable gait motion, Center of Mass (CoM) and Ground Projection of Center of Mass (GCoM) must be in a straight line where the GCoM must always be within the foot sole area, as shown in Fig. 5. If GCoM is outside of the foot sole area, the robot will lose balance and fall down.
Notice that when swinging one leg, the waist moves to be on top of another leg in order to shift CoM position so that the CoM is centered with the GCoM. These movements bring together the whole robot trunk to left and right simultaneously. Therefore, to safely navigate the biped locomotion in a humanoid robot, it is necessary to consider the trunk movement of the robot body. In this study, the trunk movement is considered as a parameter value r, as shown in Fig. 5, which is taken as the distance from waist-joint to hip-joint. Eventually, this kind of walking pattern delays the walking speed. Moreover, joint structure design in robots does not permit flexible movement like that of human being. Indeed, one motor only can rotate in one direction. Even by reducing reduction-ratio can increase the motor rotation, it will eventually reduce the torque output which is not desirable for real-time operation.
Therefore, instead of stabilization issue that have been presented in many research, analysis to increase walking speed is necessary in biped locomotion so that the robots can move faster without reducing the reduction-ratio at the motor system, which will jeopardize their ability to perform tasks. Furthermore, not many works have been reported regarding analysis of biped walking speed.

4.2 Analysis of Biped Walking Speed

We have identified that five main tasks need to be solved in the contact-based navigation system for biped robots: searching, self-localization, correction, obstacle avoidance, and object handling. On top of these tasks, walking locomotion is the basic motion that supports the tasks. It is therefore an efficient biped locomotion strategy in walking motion is required in the navigation system. For the sake of navigating a biped humanoid robot, the objective is to generate efficient gait during performing tasks and maintain in stable condition until the tasks are completed.

The efficiency in biped robots is normally related with how fast and how easy the tasks can be completed. In fact, a method to control sufficient walking speed in conjunction with the biped gait trajectory is inevitably important. This is because in real-time application, the robots are likely to be required to walk faster or slower according to situation that occurred during the operation. It is therefore we need to identify parameters to control walking speed in biped locomotion. Previously, several studies have been reported related with walking speed of biped robot. For example Chevallereau & Aoustin (Chevallereau & Aoustin, 2001) have studied optimal reference trajectory for walking and running of a biped robot.

Furthermore, Yamaguchi (Yamaguchi et al., 1993) have been using the ZMP as a criterion to distinguish the stability of walking for a biped walking robot which has a trunk. The authors introduce a control method of dynamic biped walking for a biped walking robot to compensate for the three-axis (pitch, roll and yaw-axis) moment on an arbitrary planned ZMP by trunk motion. The authors developed a biped walking robot and performed a walking experiment with the robot using the control method. The result was a fast dynamic biped walking at the walking speed of 0.54 s/step with a 0.3 m step on a flat floor. This walking speed is about 50% faster than that with the robot which compensates for only the two-axis (pitch and roll-axis) moment by trunk motion. Meanwhile, control system that stabilizes running biped robot HRP-2LR has been proposed by Kajita (Kajita et al., 2005). The robot uses prescribed running pattern calculated by resolved momentum control, and a running controller stabilizes the system against disturbance.

In this research, we focus in developing efficient walking locomotion by improving the walking gait velocity and travel distance. This analysis employs the humanoid robot Bonten-Maru II as an analysis platform. Eventually, it is easy to control the walking speed by reducing or increasing the reduction-ratio at the robot joint-motor system. However, in real-time operation it is desirable to have a stable and high reduction-ratio value in order to provide high torque output to the robot's manipulator during performing tasks, such as during object manipulation, avoiding obstacle, etc. Therefore the reduction-ratio is required to remain always at fixed and high value.

4.2.1 Selection of Parameters

The main consideration in navigating a biped humanoid robot is to generate the robot's efficient gait during performing tasks and maintain it in a stable condition until the tasks are completed. The efficiency in biped robots is normally related with how fast and how easy the tasks can be completed. In this research, to increase walking speed without changing the reduction-ratio, we considered three parameters to control the walking speed in biped robot locomotion:

1) Step length; s
2) hip-joint height from the ground; h
3) Duty-ratio; d

Figure 6 shows initial orientation of Bonten-Maru II during motion mode which also indicate the step length and hip-joint height of the robot. The step-length is the distance between ankle-joints of a support leg and a swing leg when both of them are settled on the

ground during walking motion. The hip-joint height is the distance between intersection point of hip-joint roll and pitch to the ground in walking position. Meanwhile, duty-ratio for biped robot mechanism is described as time ratio of one foot touches the ground when another foot swing to transfer the leg in one cycle of walking motion.
In biped gait motion, two steps are equal to one cycle (refer Fig. 5). Figure 6 also shows link dimension of the Bonten-Maru II body and structure of the leg. The link parameters at the legs are used in calculations to define hip-joint height and maximum step length by geometrical analysis. Link parameters of the legs were calculated geometrically to define relation between step-length and hip-joint height. From the geometrical analysis, relation between the step-length and the hip-joint height is defined in Table 2.
At joint-motor system of Bonten-Maru II, maximum no-load rotation for the DC servomotor at each joint is 7220 [rpm]. This rotation is reduced by pulley and harmonic drive-reduction system to 1/333, in order to produce high torque output during performing tasks. We considered that the robot required high torque to perform tasks; therefore we do not change the reduction-ratio, which is 333:1. Eventually, these specifications produced maximum joint angular velocity at 130 [deg/s]. However, for safety reason, the joint angular velocity at the motor was reduced to 117 [deg/s].
The step time can be adjusted in the robot control system easily. However, if the step time is too small in order to increase walking speed, the robot motion becomes unstable. Moreover, the maximum step length performed becomes limited. In current condition, the step time for Bonten-Maru II to complete one cycle of walking is fixed between 7~10 second at maximum step length 75 [mm]. The duty-ratio *d* is increased gradually from 0.7 to 0.85.

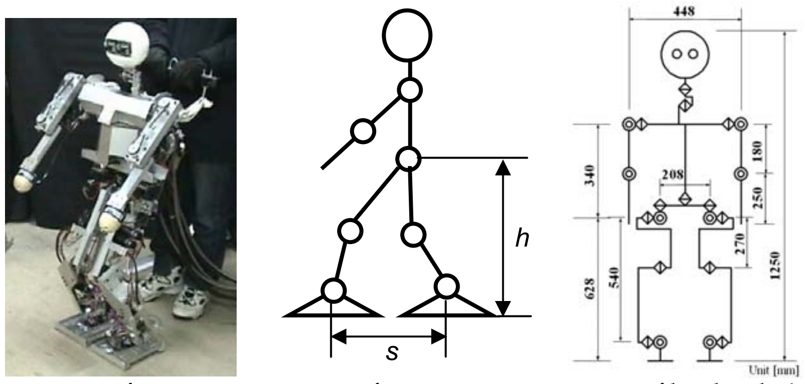


Fig. 6. Orientation of Bonten-Maru II to perform motion, parameters of hip-height *h* and step length *s*, and diagram of link dimensions.

Hip-joint height [mm]	Max. step length in 1 step[mm]	Max. step length in 1 cycle [mm]
h_1=468	350	700
h_2=518	300	600
h_3=568	200	400

Table 2. Relationship of step length against hip-joint height at Bonten-Maru II.

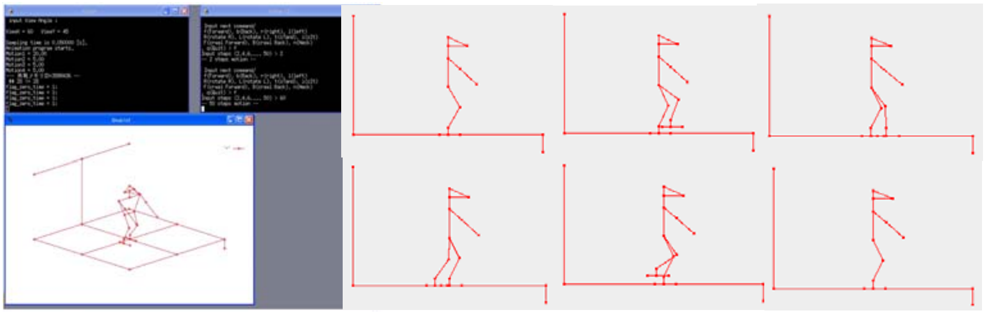


Fig. 7. Simulation by animation presents robot's trajectory in biped walking motion.

4.2.2 Simulation Analysis

A simulation analysis of the robot walking velocity using animation that applies GnuPlot was performed based on parameters condition explained at previous section. The time for one circle of walking gait is initially fixed at 10 second. Figure 7 displays the animation screen of the robot's trajectory, which features a robot animation performing walking motion. Each joint's rotation angles are saved and analyzed in a graph structure. Based on the joint angle, angular velocity of each joint was calculated.

For example, Fig. 8 shows joint angle data for right leg joints when performing 10 steps walk at condition: *h*=518 [mm], *s*=100 [mm] and *d*=0.7. From the angle data, angular velocity for each joint was calculated and presented in Fig. 9. The first and last gait shows acceleration and deceleration of the gait velocity. The three steps in the middle show maximum angular velocity of the legs joint. Basically, in biped robot the maximum walking gait velocity is calculated from maximum joint angular velocity data by defining minimum step time for one gait. Eventually, by applying the same parameter, even if time for one step is initially different; the final joint angle obtained by the robot is same. Hence, in this analysis we can obtain the minimum step time in one step from the maximum joint angular velocity data that the initial step time was 10 seconds. Basically, the minimum gait time in one step is satisfying following equation:

$$t_{\min} < \frac{v_{\theta\max} \times 10}{V_{\theta\max}} \, . \tag{32}$$

Here, $V_{\theta\max}$ is the maximum joint angular velocity at the motor, t_{min} is minimum time for one step, and $v_{\theta max}$ is maximum joint angular velocity in each gait. Finally, the maximum walking gait velocity w_{max} is defined by dividing length s with minimum step time t_{min} in each gait, as shown in following equation.

$$w_{\min} = \frac{s}{t_{\min}} \tag{33}$$

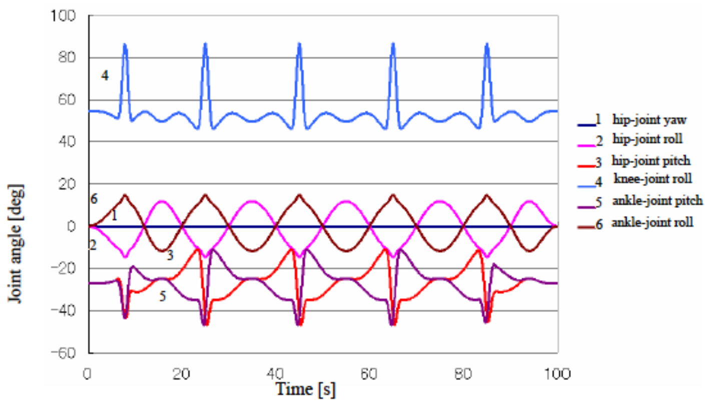


Fig. 8. Graph of joint rotation angle at right leg.

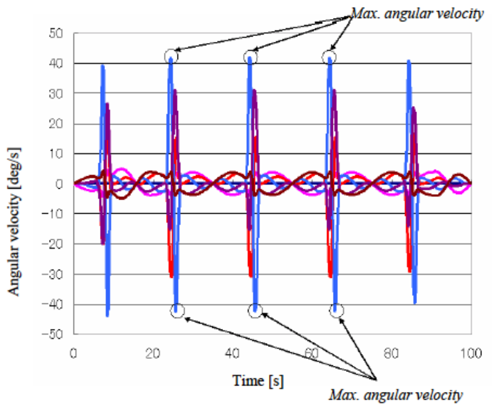


Fig. 9. Graph of angular velocity of joint rotation at right leg.

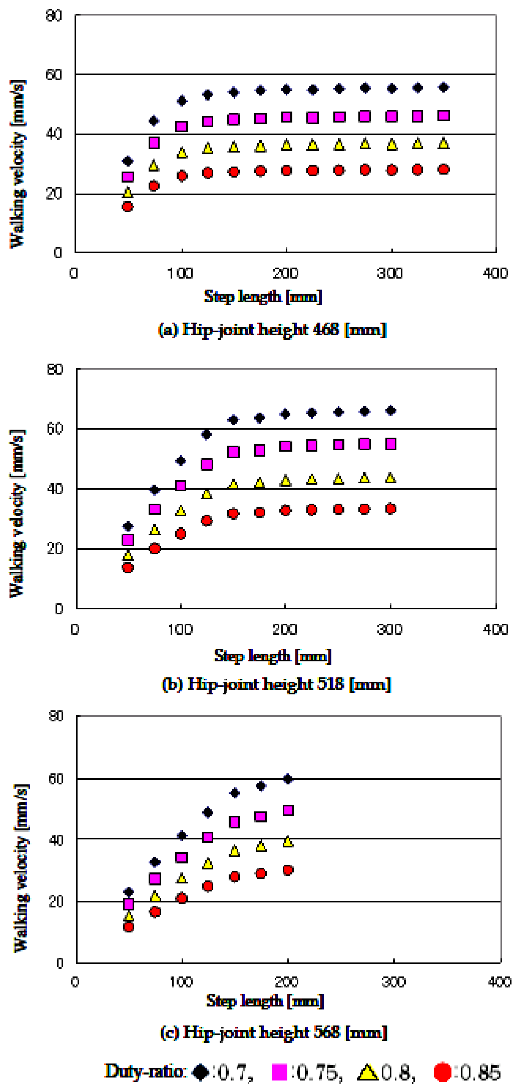


Fig. 10. Analysis results of maximum walking velocity at each gait.

4.2.3 Simulation Results

Simulation results of walking gait velocity at each parameters value are compiled in graphs as shown in Fig. 10(a), (b) and (c). According to these graphs, from the relation of walking velocity and step length, the walking velocity was maintain nearly at constant value when it reached certain step length. Moreover, in relation of step length and hip-joint height, the higher hip-joint position is providing wider step length to perform better walking distance.

At this point, lower duty-ratio shows the best results in relation of the hip-joint height and the step length for higher walking gait velocity, as shown in Fig. 10(b), where the low duty-ratio shows high walking velocity in relationship between the hip-joint-height and the step-length. It means by shorten the time for the support leg touching the ground will urge swing leg to increase its speed to complete one walking cycle, thus increase the walking velocity. At the same time, by choosing suitable step-length and hip-joint-height parameters, travel distance in each step can be improved. This analysis results revealed that it is possible to control biped walking speed without reducing the reduction-ratio at the joint-motor system. From the simulation results, we can conclude that lower duty-ratio in suitable hip-joint height comparatively provided higher walking gait velocity. For Bonten-Maru II, the maximum walking gait velocity was improved from 30 [mm/s] to 66 [mm/s], which is about two times better than current walking velocity. At this time the hip-joint height is 518 [mm] and the time for one step is 4.5 seconds.

4.2.4 Experiment with Bonten-Maru II

We conduct experiments with the biped humanoid robot Bonten-Maru II. The parameter values that revealed the best result in simulation were applied, in comparison with current walking condition. Figures 11 and 12 respectively show photograph of the actual robot's walking motion in each experiment, which also indicate the parameter values applied. Travel distance was measured during the experiments. The experimental results show that by applying the best parameters value obtained in the simulation results, the walking speed was improved. At the same time, the travel distance is longer about three times compared with current condition.

This result reveals that the travel distance was improved in conjunction with the improvement of walking speed in the biped humanoid robot. The robot performed biped walking in smooth and stable condition. The experiments utilizing real biped humanoid robot based on simulation results shows that the robot's travel distance during walking was improved about three times better than current walking condition. This analysis proved that it is possible to improve walking speed in stable biped locomotion without reducing the reduction-ratio. This analysis results contributes to reliable biped locomotion during performing tasks in the humanoid robot navigation system.

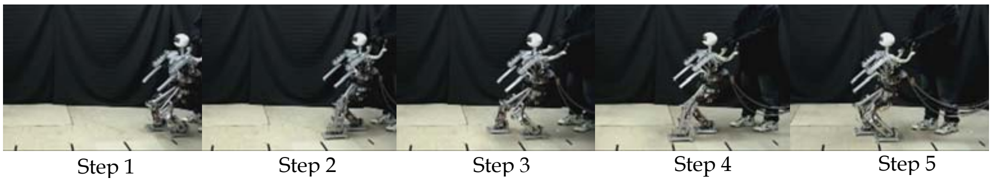


Fig. 11. Humanoid robot performs biped walking applying the best parameters value from simulation results: h=518 [mm], s=200 [mm] and d=0.7, time per step 4.5 sec.

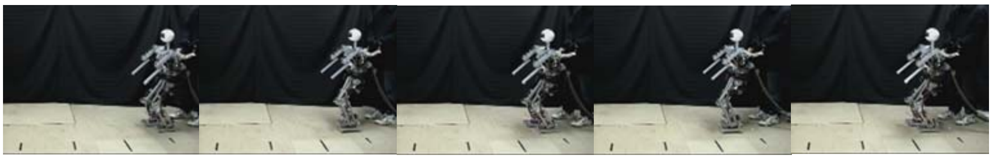


Fig. 12. Humanoid robot performs biped walking applying current parameters value: h=568 [mm], s=75 [mm] and d=0.8, time per step 2.5 sec.

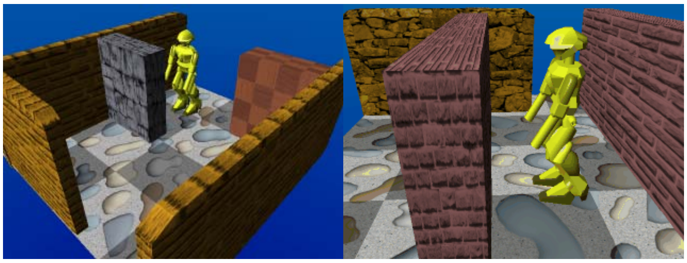


Fig. 13. Humanoid robot operates in confined space.

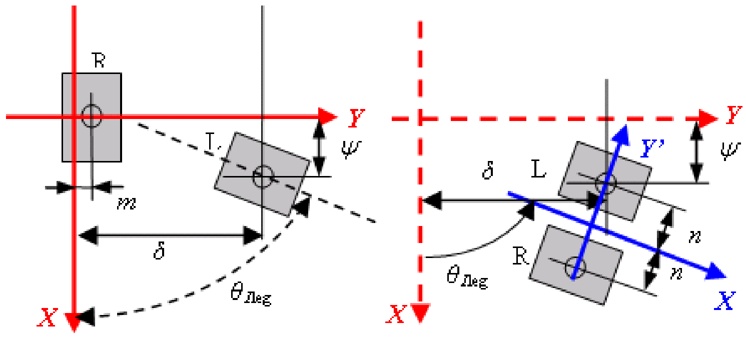
4.3 Analysis of Yawing Motion

To effectively operate in unseen environment, it is necessary for humanoid robot to have skills to operate in limited space, such as in a small room or a tunnel as shown in Fig. 13. In navigation within confined space, it is critical for biped robots to change their orientation within a very limited space in order to change locomotion direction. Motivated by this situation, we analyzed humanoid robot with its rigid body structure to define biped locomotion strategy to perform yawing movement towards operation in confined spaces.

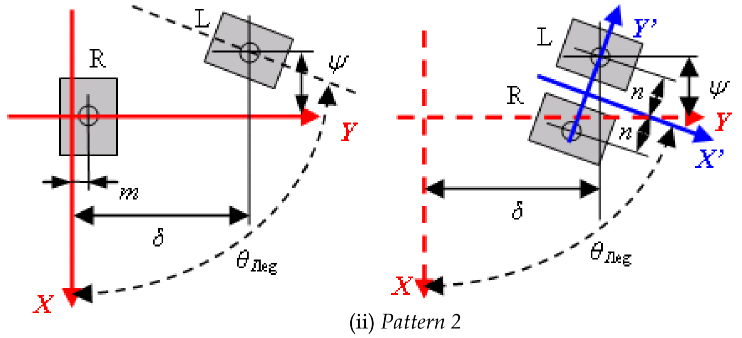
In this study, we identified two types of yawing movement to change the robot orientation at this situation. The first movement, so called *pattern 1*, is when the robot changed its orientation to the front-left or front-right area. The second movement called *pattern 2* is when the robot turn to the back-left or back-right area. Geometrical analyses of these two yawing patterns for left side movements are illustrated in Fig. 14. Here, m, n, δ and ψ are dimensional results from calculations to define each leg's position which taking hip-joint-height, length of the leg's links and maximum step length as parameter values. The xy-axes gives reference coordinates before rotation, while the $x'y'$-axes gives coordinates after rotation is completed. In order to change the robot's orientation, rotation of hip-joint yaw is the key-point, as shown in Fig. 15.

By solving inverse kinematics calculation to define joint rotation angle θ_{1leg} as presented in previous section, yawing rotation of the leg was performed. The robot rotation angle was decided from *0* degree to *90* degree for both patterns. Meanwhile target position of leg's end-point at X-Y axes plane defined by interpolation formulations distinguishing leg's movements of the *pattern 1* and *pattern 2*. Motion planning for *pattern 1* and *pattern 2* are shown in Fig. 16. For example, in the case of rotating to left side, at first the left leg's hip-joint yaw will rotate counterclockwise direction to θ_1 (see Fig. 15). At the same time, the left leg follow along an ellipse trajectory in regard to z-axis direction to move the leg one step.

This stepping motion is performed by given the leg's end point position to the robot's control system so that the joint angles of θ_{1leg}~θ_{6leg} could be solved by inverse kinematics.
The left leg position is defined by interpolation of the leg end point from its initial position with respect to the *xy*-axes position at a certain calculated distance (see Fig. 14). At this time the right leg acts as the support axis. Then, the robot corrects its orientation by changing the support axis to the left leg, while the left leg hip-joint yaw reverses the rotation to clockwise direction to complete the motion. These yawing movements minimize the rotation space of the robot, which consequently minimize the possibility of collision with objects around the robot. Furthermore, it is easier in control whereby only one leg performing rotation, while the other leg functioned as support axis. The rotation angle at hip-joint yaw also permits wide rotation range.



(i) *Pattern 1*



(ii) *Pattern 2*

Fig. 14. Geometrical analysis of the robot's foot-bottom position during yawing movements of *pattern 1* and *pattern 2*.

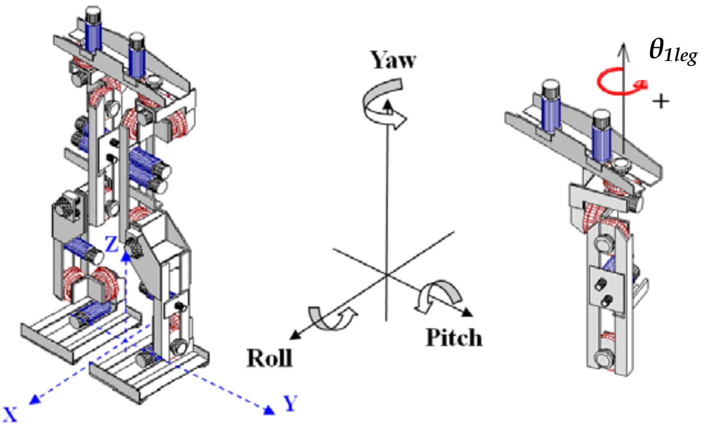


Fig. 15. Leg structure and rotation of hip-joint yaw.

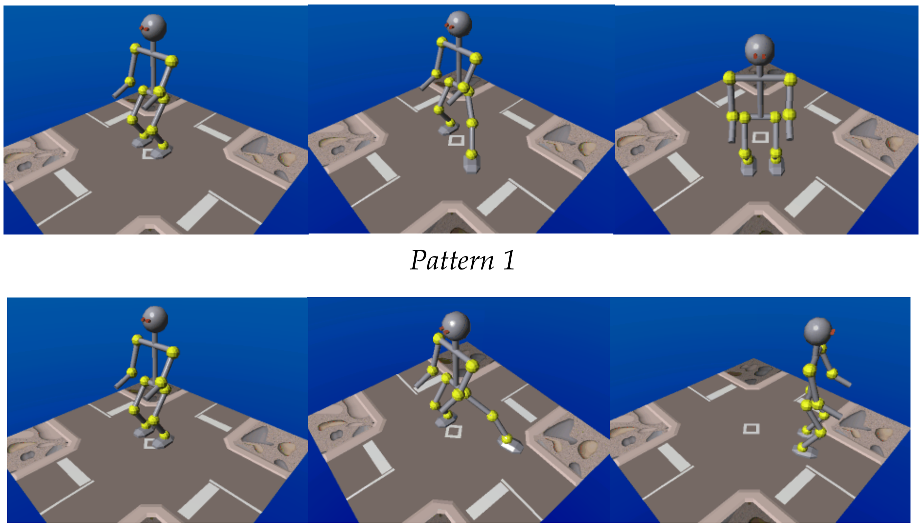
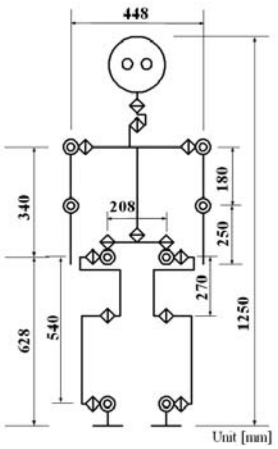


Fig. 16. Motion planning of yawing motions in biped humanoid robot.

5. Analysis of Collision Avoidance

Eventually, to safely and effectively navigate robots in unseen environment, the navigation system must feature reliable collision checking method to avoid collisions. The purpose of analyzing collision checking in this research is to define collision free area based on the arm's detection area in searching tasks. The aim is to avoid collision with external objects so that the robot can perform walking motions safely in the navigation tasks. In conjunction with this collision free area, a suitable step-size of the robot's foot during walking motion is defined. We assumed that no objects were detected in the searching process, so that we can

calculate maximum size of the collision free area. Consequently, when object is detected in searching process, the robot control system can recalculate the collision free area automatically based on the object's distance from robot.
Since the proposed navigation system in this research only depends on contact interaction of both arms to realize environment conditions, calculation of arm's links parameters and detection radius area in correlation with biped walk characteristics are important. Therefore, the collision checking method proposed in this research is considered the link parameters of the robot's arms, and trunk movement during biped walk motions as parameter values. The collision free area is used to calculate maximum biped step-length when no object is detected. Consequently the robot control system created an absolute collision free area for the robot to generate optimum biped trajectories. In case of object is detected during searching motion, the robot arm will touch and grasp the object surface to define self-localization, and consequently optimum step-length is refined.
In this chapter, we present analyses of collision checking using the robot arms to perform searching, touching and grasping motions in order to recognize its surrounding condition. The collision checking is performed in searching motion of the robot's arms that created a radius of detection area within the arm's reach. Based on the searching area coverage of the robot arms, we geometrically analyze the robot biped motions using Rapid-2D CAD software to identify the ideal collision free area. The inputs are humanoid robot Bonten-Maru II link dimensions and joint rotation range as shown in Fig. 17. The outputs are radiuses of both arm rotations indicates detection area which taking centre of rotation at shoulder and elbow joints and absolute collisions free trajectory area of the robot legs.



Joint (Axis)	Range of rotation angle [deg.]
Right shoulder (roll)	-135 ~ 30
Left shoulder (roll)	-30 ~ 135
Elbow (roll)	-90 ~ 90
Hip (yaw)	-90 ~ 90
Right hip (roll)	-90 ~ 25
Left hip (roll)	-25 ~ 90
Hip (pitch)	-130 ~ 45
Knee (pitch)	-20 ~150
Ankle (pitch)	-90 ~ 60
Right ankle (roll)	-90 ~ 20
Left ankle (roll)	-20 ~ 90

Fig. 17. Humanoid robot Bonten-Maru II link dimensions and joint rotation range.

5.1 Methodology

The purpose of analyzing collision checking in this research is to define collision free area based on the arm's detection area in searching tasks, and calculate suitable step-length of the robot leg in biped walking motions. The strategy will help humanoid robot to reduce possibility of collisions with external objects, at the same time help the robot control system to define optimum biped gait trajectories in the proposed navigation system.

In the analysis, at first we assumed that no objects were detected in the searching process, so that we can define the maximum size of collision free area and consequently calculate the maximum step-length of the robot leg during walking to forward, backward and side directions.. Meanwhile, when object is detected in searching process, the robot can measure the collision free area based on the acquired object's distance and the specified maximum step-length. Accordingly, optimum step-length can be defined automatically in the robot control system.
Since the proposed navigation system in this research only depends on contact interaction of both arms to realize environment conditions, we must consider link dimensions of the robot's arm and the leg as parameter values. Furthermore, consideration of optimum arm's motions during searching tasks and its correlation with biped walk characteristics is also important. For Bonten-Maru II, we utilized parameters of link dimensions at the arms and legs as shown in Fig. 17. Meanwhile, motion of arms during searching tasks is classified in two patterns: First is the motion where rotation is centered at elbow joint, which is during searching object at the robot front area, and second is the motion where rotation is centered at shoulder joint, which is during searching object at robot side area.

5.2 Condition I: When No Object Detected

Initially, we perform analysis assuming that no object is detected in searching area so that we can identify the optimum collision free area and maximum step-length. The proposed arm rotation as explained in previous section created circular shape areas as shown in Fig. 18, which radius r_1 is the arm's upper link, and r_2 is additional of lower and upper arm links parameters. These radiuses are actually replicating the arm's end-effector position during searching tasks. Based on the searching area coverage of the robot arms, we geometrically analyze the robot biped motions using Rapid-2D CAD software to identify the ideal collision free area. The relation of hip-joint height from ground h and step-length S in biped walking robot, as shown in Fig. 6, is considered in the analysis. In our research utilizing humanoid robot Bonten-Maru II, the robot hip-joint height during walking motion was fixed at h=518 [mm]. Meanwhile, upper and lower link parameters of the arms are L_1=180 [mm], L_2=250 [mm], respectively.
Figure 19 shows analysis result using Rapid-2D CAD software of the robot performing searching motion and biped walking to various directions. In this figure, at first the robot starts the motion by searching process. Then it walk to forward, backward and side-step directions, turn to left and right directions at 45 degree, and turn back to right and left directions at 70 degree. We considered the body trunk movements, foot sole positions, and the arms and legs rotation range at all walking directions to define the collision free area. Finally, we defined a rectangular shaped collision free area as shown in the Fig. 6. This figure also indicated radius of collision checking area replicating the arms end-effector position, and also foot sole area.
Eventually, as shown in Fig. 20, we can still see some areas within the rectangular shaped collision free trajectory area are out of the coverage of the collision checking radius. However, these areas are actually mostly covered when the robot start moving, for example as shown in Fig. 20 (right) where the robot performed side step motion to right-side direction. Indeed, it called absolute collision free trajectory area because the area is covered almost all walking directions where the robot can safely perform biped walking trajectories to satisfy tasks in the proposed navigation system. From the analysis result, we defined the

maximum step-length for Bonten-Maru II when no object was detected: 75 mm/step for walking to forward and backward, 100 mm/step for walking to side-step, and 75 mm/step for walking to tangential directions.

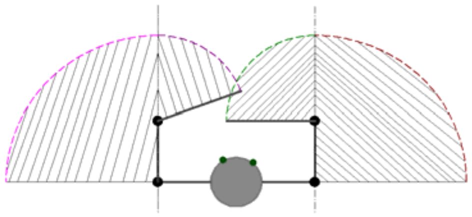
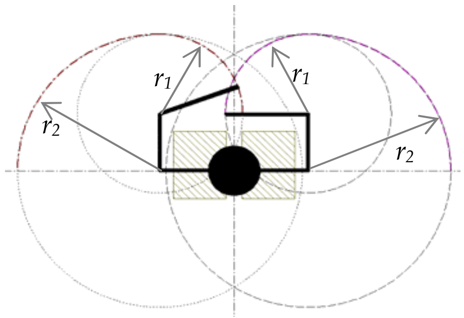



Fig. 18. (Left) Actual searching coverage area of both arms. (Right) Radiuses of both arm rotations indicates detection area taking centre of rotation at shoulder and elbow joints.

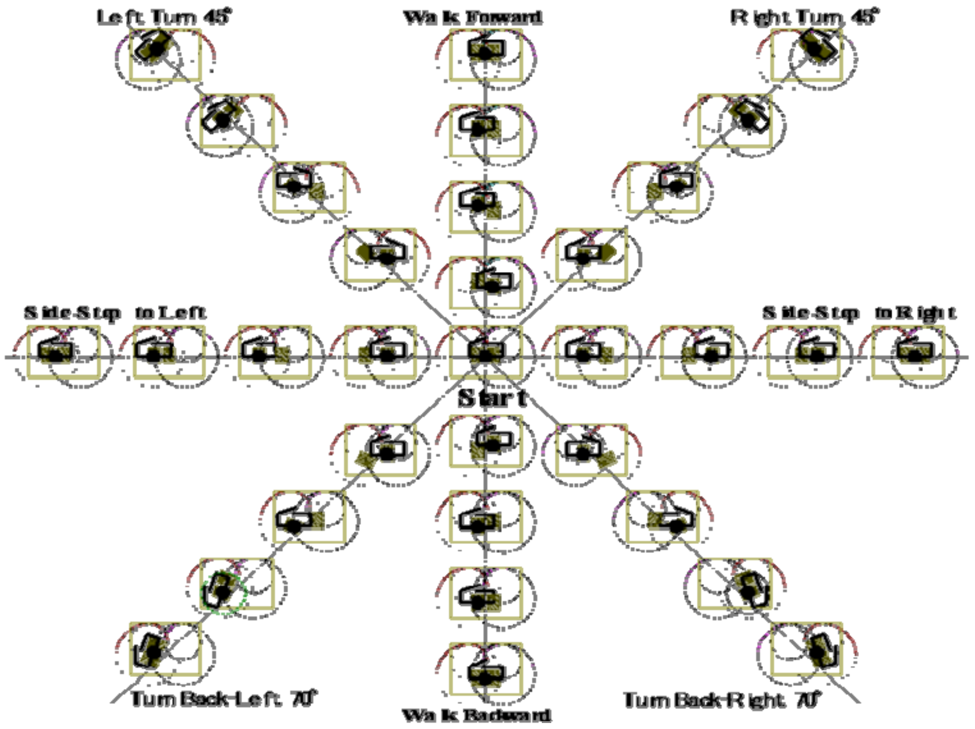


Fig. 19. Analysis of collision checking for humanoid robot Bonten-Maru II in walking forward, backward, side to left and right, and tangential directions using Rapid-2D software.

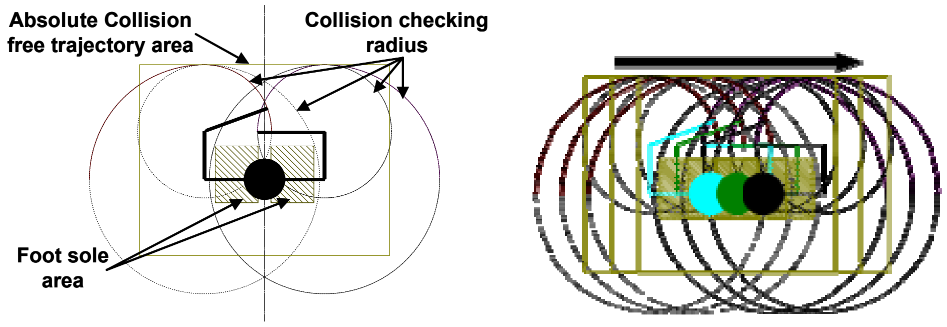

Absolute Collision free trajectory area
Collision checking radius
Foot sole area

Fig. 20. (Left) Absolute collisions free trajectory area and collision checking parameters. (Right) Collision checking areas in walking side-step to right direction.

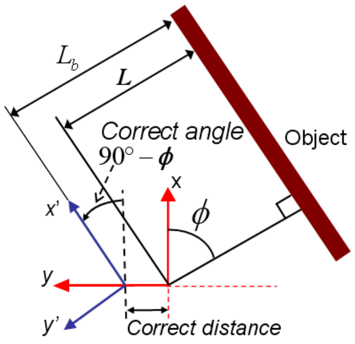


Fig. 21. Robot orientation after grasping process.

5.3 Condition II: When Object Detected

In situation where object is detected during searching tasks, the robot arm will grasp the object surface and define self-localization. The control system will measure the object orientation and then consequently calculate the robot distance and angle. Figure 21 shows the geometrical analysis of the robot position and angle after the grasping process. During grasping process, the arm's end-effector position data were defined. The position data are calculated with the least-square method to result for a linear equation, as shown in (34) where a is the slope of the line and b is the intersection at y-axis. A straight line from the reference coordinates origin that is perpendicular with (34), which describes the shortest distance from robot to wall, is defined in (35). Consequently, the intersection coordinate at the x-y axes plane is defined in (36). Here, grasping angle ϕ is an angle from the x-axis of the robot reference coordinates to the perpendicular line of (35). Finally, the distance of the robot to object L and grasping angle ϕ are shown in (37) and (38), respectively.

$$y = ax + b \tag{34}$$

$$y = -\frac{1}{a}x \tag{35}$$

$$\left(C_x, C_y\right) = \left(-\frac{ab}{a^2+1}, \frac{b}{a^2+1}\right) \tag{36}$$

$$L = \frac{b}{\sqrt{a^2+1}} \tag{37}$$

$$\phi = \tan^{-1}\left(-\frac{1}{a}\right) \tag{38}$$

For collision checking, the size of collision free area is refined automatically in the robot control system based on parameter value of shortest distance L. Consequently, a parameter value L_b which considered safety distance between the robot to the object during walking locomotion is defined based on link parameters of the arm. Accordingly, the step-length s of the robot during biped walking is recalculate using shortest distance L and grasping angle ϕ as indicated in (38).

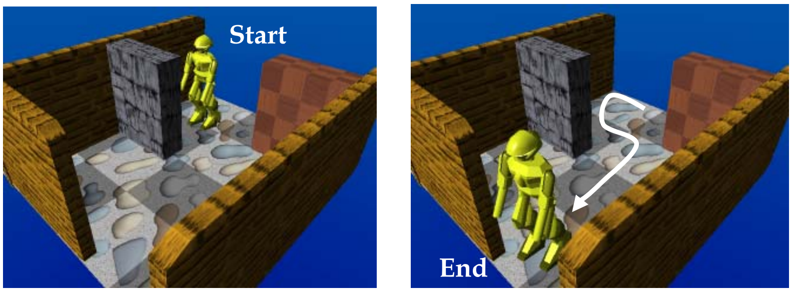


Fig. 22. Layout of Humanoid robot operates in small room with wall and obstacle.

6. Experiment

Verification experiments using humanoid robot Bonten-Maru II to operate in a room with walls and obstacles was conducted. The walls and obstacles were arranged as shown in Fig. 22. In this experiment, the robot visual sensors are not connected to the system. Therefore the robot locomotion can only rely on contact interaction of the arms that are equipped with force sensors. A human operator acts as the motion instructor and controls the robot motions. The operator has prior knowledge of the room arrangement but does not know the exact starting position of the robot.

6.1 Motion Planning

Observation of human behavior during operation in a dark room shows that at first the human subject seems to search and identify the nearest object to recognize his position in the room. Inspired by this scenario, the humanoid robot starts the navigation process by searching for the nearest object to define self-localization so that it can recognize its position and orientation in the environment where the operation takes place. Continuously the robot generates suitable locomotion based on contact information; as described by the proposed algorithm explained in the previous section.

Lastly, the navigation of a walking robot requires accurate collision avoidance parameters.

This ensures the safety of robot when it performs locomotion in the navigation tasks. As mentioned in section 4, the robot trunk movement parameter *r* is one of the parameters which need to be considered. The arm detection area which is equal to the length of arm L_t also needs to be considered as a parameter. The robot travel distance (step length and quantity of steps) in walking motions is controlled based on these parameter values. For forward and backward walking motion, travel distance must be less than the value of L_t. Whereas for the side-stepping motion and tangential walking during the correction process, the travel distance is fixed to be less than parameter value of $L_t + r$.
In order to explain motion planning in this experiment, we simulate room with walls and obstacles to overview the path planning of the proposed method. Figure 23 shows top view of the robot orientation while it performs locomotion in the simulation room. The robot starts the locomotion by searching for nearest wall using right arm. If no wall is detected, robot will walk side-step to its right direction and repeats the searching process until the arm detects a wall. When the wall is detected, the robot's arm will grasp the wall surface to obtain the wall's orientation. Base on the obtained wall orientation, the robot control system could recognize its position and orientation against the wall.

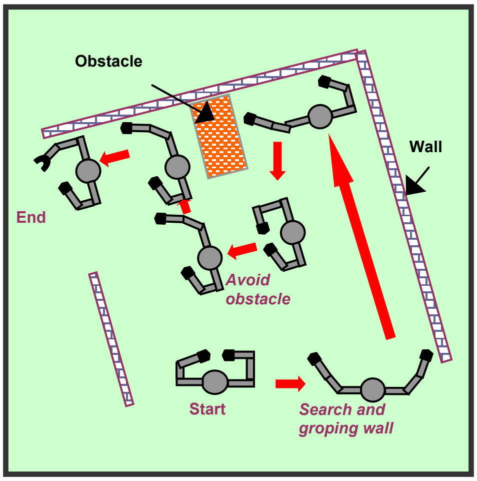


Fig. 23. Motion planning in navigation experiment.

Next, the robot performs locomotion to correct its orientation, at first by correcting its position. After correction of position, the robot's left arm will check the existence of any obstacles at the targeted correction of angle area. If no obstacle is detected, the robot will continue correcting its orientation by changing its direction angle. If the obstacle is detected, the robot will perform trajectories to avoid the obstacle. The robot will repeat the whole process again until it reaches to the desired target point. In this research, the robot control system is intelligently classified each process so that the robot could effectively perform suitable locomotion according to the surrounding conditions.

6.2 Motion Algorithm

In the proposed humanoid robot navigation strategy, we designed a motion algorithm consisting of searching and detecting object, self-localization by grasping, correction of locomotion direction, and obstacle avoidance (Hanafiah et al., 2008). Figure 24 shows the flowchart of the motion algorithm. The algorithm comprises formulations of kinematics, interpolation of manipulator's end-effector, and force-position control in order to generate trajectory for each robotic joint.

First the robot searches the nearest object using both arms. When no object was detected, the robot walked side-step to its right direction. In this navigation algorithm, navigation priority was given to right-side direction so that optimum motions could be achieved during navigation tasks. Then the robot repeats again the search and detection tasks. When the object was detected, the robot grasped its surface to define self-localization. Self-localization here means calculation results of distance and orientation of each humanoid robot's joints towards the object. Then the robot checked for any obstacle existence in the correction area using its left arm. When no obstacle was detected, the robot corrected its locomotion direction by performing yawing motions with both legs. Then the robot repeats again the searching process. The robot continued its locomotion while touching and avoiding objects in its path. When it was necessary, the robot changed its direction by turning right or left.

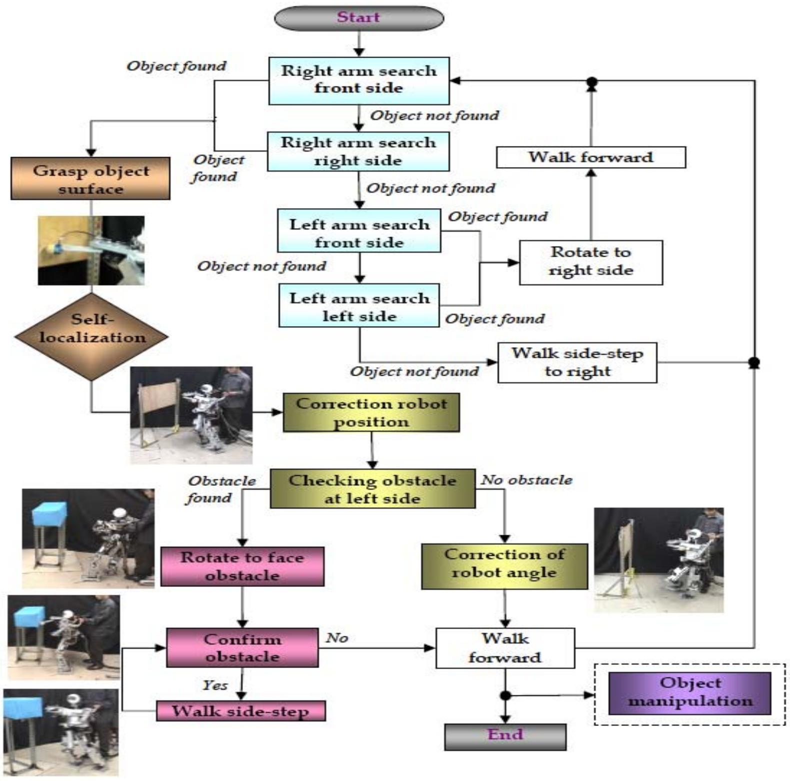


Fig. 24. Layout of motion algorithm in contact interaction-based navigation.

6.3 Experiment Result

Since the robot's vision sensors were not connected to the system, the robot locomotion had to rely on the contact interaction of the arms equipped with force sensors. The proposed navigation algorithm was installed in the robot control system.

Figure 25 shows sequential photographs of actual robot locomotion during the experiment. At first the robot was searching for the nearest object using both arms. When the object was detected, the robot grasped the object surface to define self-localization. Then the robot corrected its locomotion direction after checking for obstacle in the correction area. The robot continued its locomotion while touching and avoiding objects along its way. When it was necessary, the robot changed its direction by turning to right and left. Finally, the robot managed to complete the navigation tasks safely and reached the target end point. Experimental results revealed good performance of the robot to realize collision free area and generate optimum biped walking.

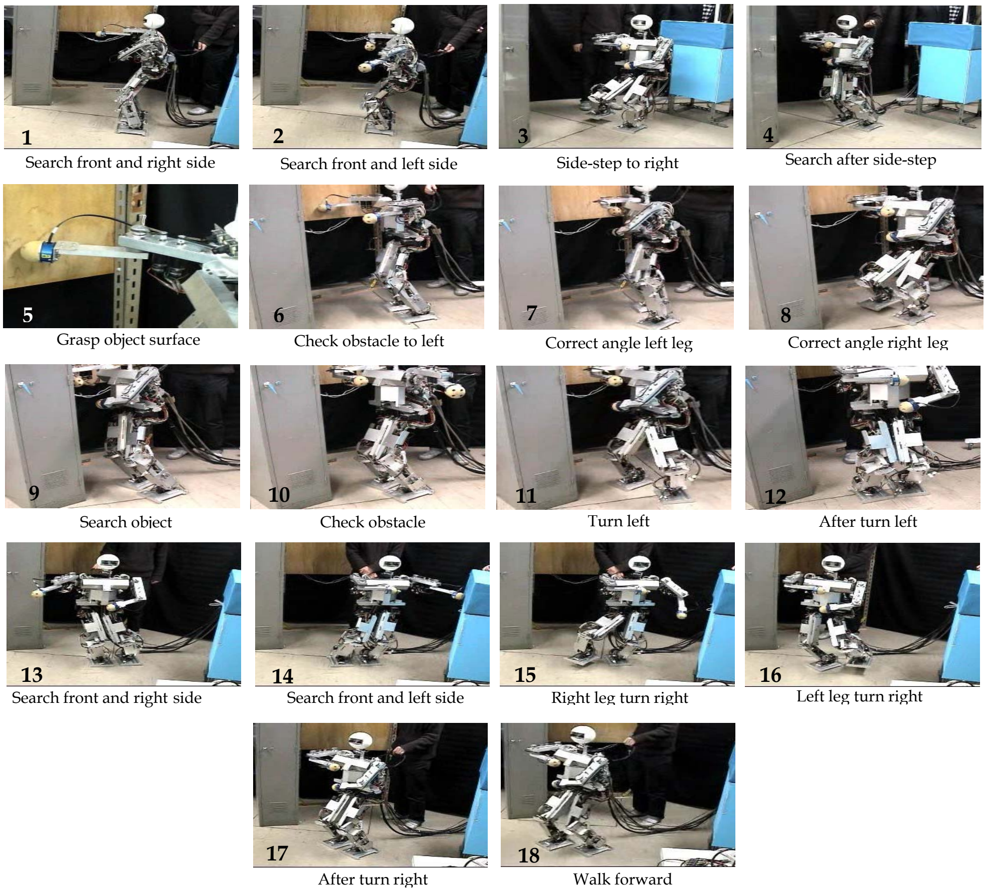


Fig. 25. Sequential photographs of humanoid robot locomotion in navigation experiment. The visual sensors were not connected to the system; therefore the robot locomotion had to rely on the contact interaction of the arms equipped with force sensors.

7. Conclusion

Research on humanoid robots in areas related with human-robot interaction has rapidly increased recently especially for application to human living environments and emergency sites. It is apparent that the environments to be shared by humanoid robots are normally dedicated to humans. In this chapter we presented analysis results of optimum biped trajectory planning for humanoid robot navigation to minimize possibility of collision during operation in unseen environment. In this analysis, we utilized 21-dof biped humanoid robot Bonten-Maru II. In this chapter, at first we analyzed the joint trajectory generation in humanoid robot legs to define efficient gait pattern. We present kinematical solutions and optimum gait trajectory patterns for humanoid robot legs. Next, we performed analysis to define efficient walking gait locomotion by improvement of walking speed and travel distance without reducing reduction-ratio at joint-motor system. Next present analyses of collision checking using the robot arms to perform searching, touching and grasping motions in order to recognize its surrounding condition.

The presented biped trajectory analysis and planning improved performance of the navigation system. This was proved by simulation and experimental results applying biped humanoid robot Bonten-Maru II during performing biped walk, side-step and yawing motions. Furthermore, analysis of efficient gait trajectory and pattern was presented in this chapter. The analysis results of the gait trajectory generation proposed an efficient gait pattern for the biped robot. Meanwhile, regarding to speed-up walk analysis, simulation results based on humanoid robot Bonten-Maru II parameters revealed that walking speed was improved by applying low duty-ratio at suitable step length and hip-joint height, whereby the walking speed increased about two times compared to normal condition. Moreover, real-time experiments utilizing real biped humanoid robot based on the simulation results showed that the robot's travel distance during walking was improved about three times longer than current walking condition. This analysis results proved that it is possible to improve walking speed in a stable biped locomotion without reducing the reduction-ratio in the robot joint-motor system. Consequently the high torque output at the robot's manipulator to conduct tasks in various motions is maintained.

We have presented collision checking method in conjunction with the motion algorithm in the contact sensing-based navigation system. The collision checking is defined by searching motions of the robot's arms that created a radius of detection area within the arm's reach, which is treated as collision free area when no object is detected. Consequently the robot control system created an absolute collision free area for the robot to safely generate trajectories in the navigation tasks. When object is detected during searching motion, the robot arm will touch and grasp the object surface to define self-localization. At this moment, the navigation map is refined and a new path is planned automatically.

The above analysis results contribute to the effort to create a stable and reliable biped walking locomotion during performing tasks in the proposed navigation system. Finally, the proposed algorithm was evaluated in an experiment with a prototype humanoid robot operating in a room with walls and obstacles. The experimental results revealed good performance of the robot locomotion in recognizing the environmental conditions and generating suitable locomotion to walk safely towards the target point. Finally, the proposed strategy was demonstrated to have good potential to support current visual-based navigation systems so that humanoids can further 'adapt' in real environment.

8. References

Chevallereau, C. & Aoustin, Y. (2001). Optimal trajectories for walking and running of a biped robot, *Robotica*, Vol. 19, Issue 5, pp. 557-569

Denavit, J. & Hartenberg, S. (1995). A kinematics notation for lower-pair mechanisms based upon matrices, *Journal of Applied Mechanics*, Vol. 77, pp. 215-221

Diaz, J. F.; Stoytchev, A. & Arkin, R. C. (2001). Exploring unknown structured environments, *Proceeding 14th Int. Florida Artificial Intelligent Research Society Conference (FLAIRS01)*, pp. 145-149

Dillmann, R. (2004). Teaching and learning of robot tasks via observation of human performance, *Journal Robotics and Autonomous Systems*, Vol. 47, Issue2-3, pp. 109-116

Gutmann, J.; Fukuchi, M. & Fujita, M. (2005). Real-time path planning for humanoid robot navigation, *Proceeding of the Int. Joint Conference on Artificial Intelligent*, pp. 1232-1238

Hanafiah Y.; Ohka, M.; Yamano, M. & Nasu, Y. (2008). Navigation strategy by contact sensing interaction for a biped humanoid robot, *International Journal of Advanced Robotic Systems (ARS)*, Vol. 5, No. 2, June 2008, pp. 151-160, ISSN 1729-8806

Hanafiah Y.; Yamano, M.; Ohka, M. & Nasu, Y. (2007). Development of a contact interaction-based navigation strategy for a biped humanoid robot, *Proceedings of 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'07)*, pp. 4241-4246, San Diego, California, USA

Hirai, K.; Hirose, M.; Haikawa, Y. & Takenaka, T. (1998). The development of Honda humanoid robot, *Proceedings of International Conference on Robotics and Automation'98*, pp. 1321-1326

Hirukawa, H.; Kanehiro, F. & Kaneko, K. (2004). Humanoid robotics platforms developed in HRP, *Journal of Robotics and Automation Systems*, Vol. 48, pp. 165-175

Ishiguro, H. (2007). Scientific issues concerning androids, *International Journal of Robotic Research*, Vol. 26, pp. 105-117

Kajita, S.; Nagasaki, T.; Kaneko, K.; Yokoi, K. & Tanie, K. (2005). A running controller of humanoid biped HRP-2LR, *Proceeding of the 2005 IEEE International Conference on Robotics and Automation (ICRA2005)*, pp. 618-624, Barcelona, Spain

Khatib, O.; Yokoi, K. & Casal, A. (1999). Robots in human environments: Basic autonomous capabilities, *Journal of Robotics Research*, Vol. 18, No. 7, pp. 684-696

Kuffner, J.; Nishikawa, K.; Kagami, S.; Kuniyoshi, Y.; Inaba, M. & Inoue, H. (2002). Self-collision detection and prevention for humanoid robots, *Proceeding of 2002 IEEE International Conference on Robotics and Automation (ICRA2002)*, Vol. 3, pp. 2265-2270

Liu, H.; Zha, H.; Chen, K. & Wang P. (2002). A new real-time collision prediction model for soccer robots, *Proceeding Korea-China Joint Workshop on Intelligent Systems*, pp.54-59, Seoul, Korea

Ogata, T.; Sugano, S. & Tani, J. (2005). Acquisition of motion primitives of robot in human-navigation task: Towards human-robot interaction based on "Quasi-Symbol", *Journal of Japanese Society for Artificial Intelligence*, Vol.20, No.3, pp.188-196

Ogura, Y.; Ando, S.; Lim, H. & Takanishi, A. (2004). Sensory-based walking motion instruction for biped humanoid robot, *Journal Robotics and Autonomous Systems*, Vol. 48, pp. 223-230

Okada, K.; Inaba, M. & Inoue, H. (2003). Integration of real-time binocular stereo vision and whole body information for dynamic walking navigation of humanoid robot, *Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems*, pp. 131-136

Sagues, C. & Guerrero, J. J. (1999). Motion and structure for vision-based navigation, *Journal Robotica*, Vol. 17, pp. 355-364

Salter, T.; Dautenhahn, K. & Boekhorst R. (2006). Learning about natural human-robot interaction styles, *Journal of Robotics and Autonomous Systems*, Vol. 52, Issue 2, pp.127-134

Seara, J.F. & Schmidt, G. (2004). Intelligent gaze control for vision-guided humanoid walking: methodological aspects, *Journal of Robotics and Autonomous System*, Vol. 48, pp. 231-248

Stasse, O.; Davison, J.A.; Sellaouti, R. & Yokoi, K. (2006). Real-time 3D SLAM for humanoid robot considering pattern generator information, *Proceeding IEEE/RSJ International Conference on Intelligent Robots and Systems 2006 (IROS2006)*, pp. 348-355, China

Perry, J. (1992). *Gait analysis: Normal and pathological function*. Slack Incorporated, pp. 4-6

Thompson, S.; Kida, Y.; Miyazaki, A. & Kagami, S. (2006). Real time autonomous navigation with a 3D laser range sensor, *Proceeding 3rd Int. Conf. on Autonomous Robot and Agents (ICARA06)*, pp. 1-8, December 2006, Palmerston North, New Zealand

Yamaguchi, J.; Takanishi, A. & Kato, I. (1993). Development of a biped walking robot compensating for three-axis moment by trunk motion, *Proceeding of 2003 EEE/RSJ International Conference on Intelligent Robots and Systems (IROS '93)*

Appendix

Σ	:	Joint-coordinate frame
$\mathbf{T}$	:	Homogeneous transformation matrix
Rot	:	Rotation transformation
Trans	:	Transfer/offset transformation
θ_i	:	Joint angle between the X_{i-1} and the X_i axes measured about the Z_i axis
d_i	:	Distance from the X_{i-1} axis to the X_i axis measured along the Z_i axis
α_i	:	Angle between the Z_i axis to the Z_{i-1} axis measured about the X_{i-1} axis
l_i	:	Distance from the Z_i axis to the Z_{i-1} axis measured along the X_{i-1} axis
s_{ij}	:	$\text{Sin}(\theta_i+\theta_j)$
c_{ij}	:	$\text{Cos}(\theta_i+\theta_j)$
atan2	:	Arc tangent of two variables
${}^0_h\mathbf{R}$	:	Matrix describes end-effector orientation with respect to reference coordinate
${}^0\mathbf{P}_h$	:	Matrix describes end-effector position with respect to reference coordinate
$P_{x,y,z}$	:	Position of end-effector at x,y,z axes
P_0	:	End-effector starting position
P_f	:	End-effector finished position
u	:	Ratio of current time and motion time

Multi-Robot Cooperative Sensing and Localization

Kai-Tai Song, Chi-Yi Tsai and Cheng-Hsien Chiu Huang
Institute of Electrical and Control Engineering, National Chiao Tung University
1001 Ta Hsueh Road, Hsinchu, Taiwan 300, ROC
E-mail: *ktsong@mail.nctu.edu.tw; u9112824@cn.nctu.edu.tw;*
hchchiu.ece90g@nctu.edu.tw

Abstract
This chapter describes a method for mobile robot localization design based on multi-robot cooperative sensing. A multi-robot cooperative localization system is presented. The system utilizes visual detection and sensor data fusion techniques to achieve mobile robot localization. The visual detection system employs a stereo vision module for both observing other robots and obtaining environmental information. Each mobile robot is able to recognize its teammates by using onboard vision system. The localization error is reduced through the proposed sensor fusion algorithm. The cooperative localization algorithm consists of two stages: serial fusion and parallel fusion. Serial fusion aims to identify the positional uncertainty of an observed robot while parallel fusion reduces its positional error based on Kalman filtering. The multi-robot cooperative localization system has been realized through the client-server architecture. Experimental results are presented to validate the effectiveness of the proposed algorithms.

Keywords: multi-robot system, mobile robot localization, cooperative localization, sensor data fusion, Kalman filtering.

1. Introduction

In recent years, multi-robot systems and distributed autonomous robotic systems have become important research areas in autonomous robotics (Parker, 1998), (Weigel *et al.*, 2002). There are increasing interests in cooperative localization and map-building using a robot team. Traditionally, robot localization and map-building are resolved by a single robot equipped with various perception sensors (Gamini Dissanayake *et al.*, 2001), (Nieto *et al.*, 2002), (Nieto *et al.*, 2003). However, a single mobile robot equipped with several types of sensors will consume much power and is more expensive to construct. A multi-robot system can overcome these drawbacks through cooperative sensing from simpler sensors and map-building will become more efficient by fusing results from each mobile robot.
Many efforts have been put to cooperative localization of mobile robots. Several researchers utilize multi-robot cooperative sensing techniques to achieve cooperative localization (Fox *et*

al., 1998), (Zhang *et al*., 2000). A Monte Carlo Localization (MCL) algorithm was developed to localize two robots equipped with laser range finders (Thrun *et al*., 2001). The presented method gathers environmental data fast and can localize individual robot using a laser scanner. A cooperative localization and mapping method was proposed for two robots to localize themselves and for map-building (Rekleitis *et al*. 2003). The key idea is that one robot carries a target to act as a landmark, and the other equipped with a laser range finder and a camera observes the former for position correction. A Kalman filter based approach was presented to simultaneous localization of a group of robots by sensing the team members and combining position information from all teammates (Roumeliotis & Bekey, 2002). The hypothesis of this method is that the robots can communicate with their teammates and measure their respective poses. A collaborative landmark localization method was presented for multiple robots equipped with vision systems to localize themselves by estimating fixed landmarks (Stroupe & Balch, 2002). The fixed landmarks are established beforehand and the robot needs to record the landmark positions. However, when the number of robots increases to more than two, information from laser scanner will not be able to distinguish each robot.

Vision-based systems have been widely exploited as a robotic perception sensor. They are also useful for distinguishing individual robot and handling the case of more than two robots. A vision-based cooperative mapping and localization method employed stereo vision to build a grid map and localize robots by features in grid map such as corners (Jennings *et al*., 1999). The key idea is to find corners in a grid map and compare these corners with *a-priori* landmarks at known positions. Hence, an accurate reference map of the environment is required in the method. A multi-robot MCL approach was presented to achieve the localization of a robot team with improved accuracy (Fox *et al*., 1999). The results show that the robots can localize themselves faster and with higher accuracy. However, although the MCL algorithm can be used for cooperative localization, it requires transmission of relatively large amount of information. On the other hand, a Kalman filtering approach can reduce not only the transmission of information but also the complexity of computing positional uncertainty. But a drawback of this approach is that the accuracy of the localization results will be decreased compared with the MCL methods. A distributed sensing method was developed based on Kalman filtering to improve the target localization (Stroupe & Martin, 2001). However, this method was only developed for recognizing and localizing a specific target. In a multi-robot cooperation system, it is desirable for each robot to recognize all of its teammates in cooperative localization. An effective algorithm is needed to fuse the observed information and reduce the sensing uncertainty.

Furtherore, when a robot team consists of more than two teammates, the problem of communication, cooperation and recognition among teammates will become increasingly important. In this chapter, we developed a multi-robot cooperative localization scheme exploiting a stereo vision system. This scheme does not restrict the number of robot in a robot team and can collect all observational information of the environment form each robot. We establish a *client-server* architecture using wireless LAN to integrate the local sensory data from each robot (a *client*) to a server. The server stores the current position of each robot in order to reduce the transmission of information between robot teammates. The proposed multi-robot cooperative localization algorithm aims to estimate the optimal position and reduce the positional uncertainty of the observed robot collaboratively through Kalman filtering. The localization method consists of two stages, serial fusion and parallel

fusion. The serial fusion is to identify the positional uncertainty of an observed robot, and the parallel fusion takes into account all of the other robots that observe other robots to reduce its positional uncertainty. Recursively, the sensory information by multiple robots can be merged using serial fusion and parallel fusion to obtain the optimal position with minimum positional uncertainty of the observed robot. It is clear that the computation of the proposed fusion algorithms is decentralized.
The stereo vision system can be exploited to construct a two-dimensional (2D) map of the environment collaboratively by multiple robots. In the current study, however, the development of map-building algorithm is not emphasized. We will show map-building results in the experimental section only. In the following sections, the presentation will focus on the design and theoretical analysis of the proposed multi-robot cooperative data fusion algorithms for localization purpose. The rest of this chapter is organized as follows. Section 2 describes the overall system architecture of the proposed multi-robot cooperative sensing system. In Section 3, the multi-robot cooperative localization scheme will be presented based on sensor data fusion technique. Section 4 presents the derivations of the Kalman filter recursive fusion formulas to obtain the fusion results. Section 5 gives some interesting experimental results of the proposed method. Finally, Section 6 summarizes the contribution of this work.

2. System Architecture

The presented multi-robot cooperative localization system is shown in Fig 1. The left block represents the environment. The middle part shows blocks of multiple robots; each of it is a *client*. Each robot captures environmental imagery using an on-board stereo camera module. Image processing algorithms are developed to observe environmental features (such as corners, edges and disparity map, etc.) and the robot's teammates from the acquired image. Stereo vision techniques are provided to find the depth information of the environment and the distance between each robot. Each robot transforms the local environmental data to a line representation through the line segment block. The upper-right block represents a server for storing the map of the environment and the current Gaussian parameters of each robot's position. In the server, all detected lines are fused through the line data fusion block, and an environmental map is maintained as line segments. The robot server receives processed local environmental data and integrates all sensory data from each robot.
Each robot utilizes odometry to calculate its Gaussian parameters of current position and updates the corresponding parameters stored in the server. Gaussian parameters of the robot are represented by its current position and the corresponding positional uncertainty. When a robot observes other robots, the serial fusion block (explained later in Section 4) estimates the other robot's position and positional uncertainty. The parallel fusion block (see Section 4) fuses via Kalman filtering the results from individual serial fusion to obtain a unique and more accurate representation. Note that if a robot does not observe other robots in its sensing region, the serial and parallel fusion processes will not be enabled. In this case, the robot only estimates and updates position and positional uncertainty of itself by odemetry. Moreover, the serial and parallel fusions are realized in each robot. In other words, the observed robot receives the serial fusion results from other robots and performs parallel fusion by itself. Therefore, the proposed multi-robot cooperative localization algorithm is decentralized. In the following sections, we will focus the discussion on the design of the presented multi-robot cooperative localization algorithm only.

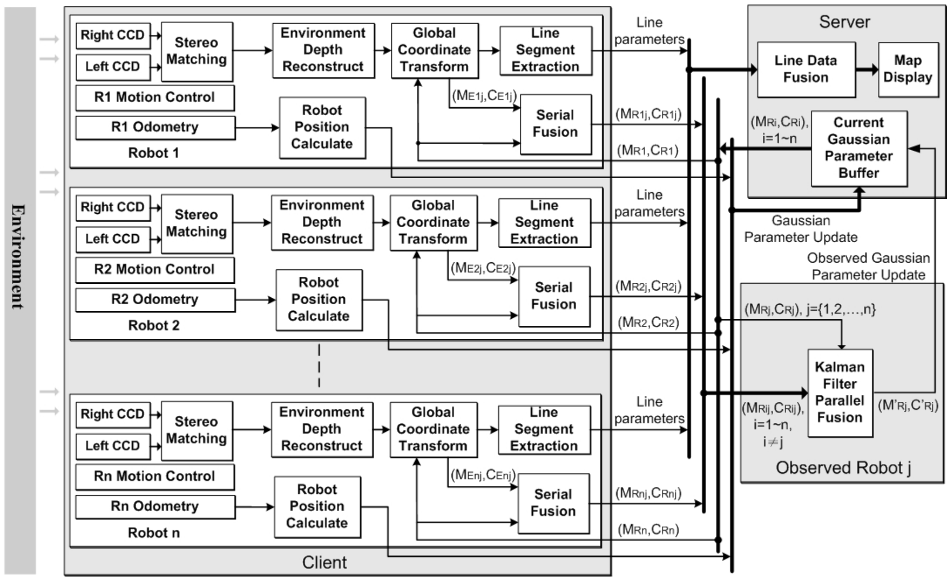


Fig. 1. System architecture of multi-robot cooperative localization design.

Remark 1: The main advantage of the discussed multi-robot cooperative localization scheme is that the computation of the localization algorithm is decentralized. However, if the Gaussian parameters of each robot are also decentralized stored in each robot, there will be a large amount of information transmission between all robot teammates due to updating their Gaussian parameters. Because information transmission is also an important factor in multi-robot systems, the external server is helpful to store the Gaussian parameters of each robot and reduce the amount of information transmission. Although the external server seems to be the disadvantage of the presented method, it helps to reduce the information transmission between robot teammates. Another drawback of the presented method is the assumption of Gaussian distribution uncertainty. Thus, in the case of multimodal distribution uncertainty, the proposed method only can provide a suboptimal solution based on a weighted least-square criterion (see Section 4.2). However, this shortcoming can be improved by extending this work by accommodating particle filters.

3. Multi-Robot Cooperative Localization Scheme

Suppose that the robot positional uncertainty can be described by Gaussian distribution, this section presents a method to reduce motion uncertainties of each robot in a robot team. Fig. 2 shows the block diagram of a recursive multi-robot localization system. M_{Ri} denotes the current position of robot i, and C_{Ri} is the corresponding positional covariance matrix. Assume that the robot j is observed by robot i, where $1 \leq i, j \leq n$ and $i \neq j$, then the estimated position M_{Eij} and the corresponding covariance matrix C_{Eij} can be generated by robot i through its on-board sensor. After merging (M_{Ri}, C_{Ri}) and (M_{Eij}, C_{Eij}) by serial fusion, the measured position and corresponding covariance matrix of robot j, (M_{Rij}, C_{Rij}), can be

obtained. The parallel fusion works to merge the current position and corresponding covariance matrix of robot j, (M_{Rj}, C_{Rj}), with all serial fusion results occurring to robot j to obtain updated parameters of robot j, (M'_{Rj}, C'_{Rj}). In the next section, we will show that the updated covariance matrix C'_{Rj} can be minimized step-by-step through parallel fusion.
It is easy to see that we separate the multi-robot cooperative localization algorithm into two stages. The first stage is serial fusion, and the second stage is parallel fusion. Table 1 tabulates the purpose and performer of serial and parallel fusions. The performer of serial fusion is the observing robot which observes another robot, and the purpose is to measure the position and the corresponding positional uncertainty of the observed robot. On the other hand, the performer of parallel fusion is the observed robot, which is observed by its teammates. The purpose is to update the current position and reduce positional uncertainty of the observed robot. Therefore, when a robot moves and observes other robots in its sensing region, serial and parallel fusions will recursively occur to localize and reduce positional uncertainties of all robots of the robot team.

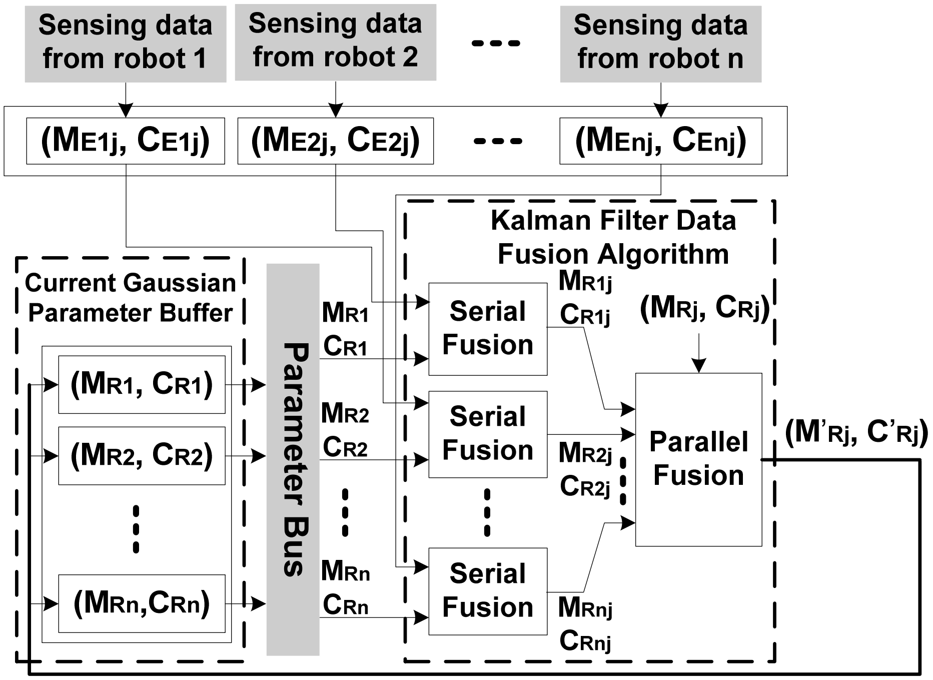


Fig. 2. Block diagram of recursive multi-robot localization.

	Purpose	Performer
Serial Fusion	Measure the position and the corresponding positional uncertainty of the observed robot.	Observing Robot
Parallel Fusion	Merge the current position and positional uncertainty of the observed robot with all serial fusion results to update the current position and reduce the positional uncertainty of the observed robot.	Observed Robot

Table 1. Purpose and performer of serial and parallel fusions.

4. Proposed Data Fusion Algorithms

In this section, a spatial uncertainty estimation method (Smith & Cheeseman, 1986) is extended to develop serial and parallel data fusion algorithms for the application of multi-robot cooperative localization. Moreover, we will also show that the parallel data fusion algorithm guarantees to provide the minimum updated covariance matrix solution via recursively Kalman filtering.

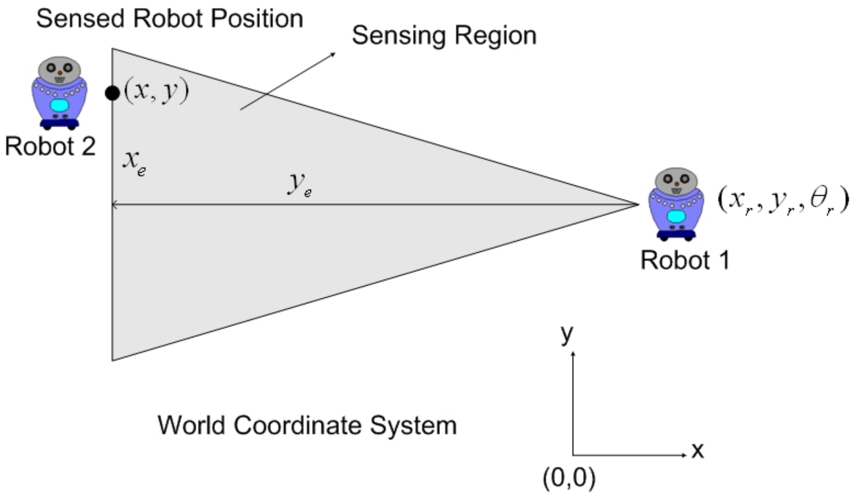


Fig. 3. Relative coordinates of two robots.

4.1 Serial fusion

The purpose of serial fusion is to measure the position and positional uncertainty of an observed robot in world coordinate. As shown in Fig. 3, a robot location is described by three parameters (x_r, y_r, θ_r) in world coordinate frame, where x_r and y_r are the *X* and *Y* coordinates of the robot, θ_r is the orientation angle of the robot. The shaded part of Fig. 3 indicates the sensing region of robot 1. x_e and y_e denote the relative position of robot 2 as observed by robot 1. If the relative orientation of robot 2, θ_e, is observed by robot 1, then the observed position and orientation of robot 2 can be estimated as:

$$x = f(M_{Ri}, M_{Eij}) = x_r + x_e \cos\theta_r - y_e \sin\theta_r, \tag{1}$$

$$y = g(M_{Ri}, M_{Eij}) = y_r + x_e \sin\theta_r + y_e \cos\theta_r, \tag{2}$$

$$\theta = h(M_{Ri}, M_{Eij}) = \theta_r + \theta_e, \tag{3}$$

where $M_{Ri} = [x_r \quad y_r \quad \theta_r]^T$ and $M_{Eij} = [x_e \quad y_e \quad \theta_e]^T$. In this presentation, the method to observe the relative position and orientation angle between robot 1 and robot 2 will not be discussed for simplicity.

Let $\overline{M}_{Rij} = [\overline{x} \quad \overline{y} \quad \overline{\theta}]^T$ and $M_{Rij} = [x \quad y \quad \theta]^T$ denote the ideal and the measured position of the observed robot, respectively; $\overline{M} = [\overline{M}_{Ri}^T \quad \overline{M}_{Eij}^T]^T$ and $M = [M_{Ri}^T \quad M_{Eij}^T]^T$ denote the ideal and the measured system parameters, respectively. By combining equations (1)-(3) into one and expanding it to a Taylor form, the following result can be obtained:

$$\begin{aligned} \overline{M}_{Rij} &= \overline{M}_{Ri} + \begin{bmatrix} \cos\overline{\theta}_r & -\sin\overline{\theta}_r & 0 \\ \sin\overline{\theta}_r & \cos\overline{\theta}_r & 0 \\ 0 & 0 & 1 \end{bmatrix} \overline{M}_{Eij} \\ &\cong M_{Rij} + J(\overline{M} - M), \end{aligned} \tag{4}$$

where J is the Jocabian matrix of M_{Rij} such that

$$J = \frac{\partial}{\partial M} \begin{bmatrix} f(M_{Ri}, M_{Eij}) \\ g(M_{Ri}, M_{Eij}) \\ h(M_{Ri}, M_{Eij}) \end{bmatrix} = \begin{bmatrix} 1 & 0 & -x_e \sin\theta_r - y_e \cos\theta_r & \cos\theta_r & -\sin\theta_r & 0 \\ 0 & 1 & x_e \cos\theta_r - y_e \sin\theta_r & \sin\theta_r & \cos\theta_r & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Based on (4), the covariance matrix of the measured position of observed robot, C_{Rij}, is given by

$$C_{Rij} \equiv E[(\overline{M}_{Rij} - M_{Rij})(\overline{M}_{Rij} - M_{Rij})^T] = J \times E[(\overline{M} - M)(\overline{M} - M)^T] \times J^T, \tag{5}$$

Because parameters M_{Ri} and M_{Eij} are obtained from different sensor systems and uncorrelated, the covariance matrix $E[(\overline{M} - M)(\overline{M} - M)^T]$ becomes such that

$$E[(\overline{M}-M)(\overline{M}-M)^T]=\begin{bmatrix} C_{Ri} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & C_{Eij} \end{bmatrix}, \tag{6}$$

where $C_{Ri} \equiv E[(\overline{M}_{Ri}-M_{Ri})(\overline{M}_{Ri}-M_{Ri})^T]$ and $C_{Eij} \equiv E[(\overline{M}_{Eij}-M_{Eij})(\overline{M}_{Eij}-M_{Eij})^T]$ are the covariance matrices of the current position of observing robot and the relative position measured by stereo vision system, respectively; $\mathbf{0}_{m\times n}$ is a m-by-n zero matrix. Based on equations (1)-(6), the serial fusion algorithm to measure the position and positional uncertainty of an observed robot in world coordinate system is summarized as follows:

Measured Position:

$$M_{Rij}=\begin{bmatrix} f(M_{Ri},M_{Eij}) \\ g(M_{Ri},M_{Eij}) \\ h(M_{Ri},M_{Eij}) \end{bmatrix}=M_{Ri}+\begin{bmatrix} \cos\theta_r & -\sin\theta_r & 0 \\ \sin\theta_r & \cos\theta_r & 0 \\ 0 & 0 & 1 \end{bmatrix}M_{Eij}, \tag{7}$$

Measured Positional Uncertainty:

$$C_{Rij}=J\begin{bmatrix} C_{Ri} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & C_{Eij} \end{bmatrix}J^T. \tag{8}$$

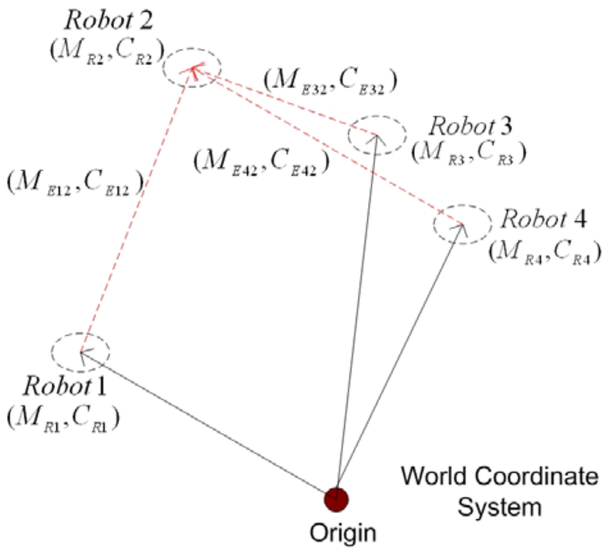


Fig. 4. Serial fusion occurs for robot 1, 3 and 4 as each of them observes robot 2.

Figure 4 illustrates the steps of serial fusion for multi-robot localization. In Fig. 4, robot 2 is observed by robot 1, robot 3 and robot 4 at the time instant k. Thus the serial fusion of robot 2 occurs for robot 1, 3, and 4 respectively. For example, the global position and uncertainty of robot 1 $(M_{R1}, C_{R1})_k$ and the estimated position and uncertainty of robot 2 from robot 1 $(M_{E12}, C_{E12})_k$ can be merged to obtain the position and uncertainty of robot 2 in world coordinate system, $(M_{R12}, C_{R12})_k$. In the same way, another two positions and uncertainties of robot 2 as observed by robot 3 and robot 4, $(M_{R32}, C_{R32})_k$ and $(M_{R42}, C_{R42})_k$, can also be obtained respectively.

4.2 Parallel fusion

Parallel fusion aims to estimate the optimal position and reduce the positional uncertainty of an observed robot using Kalman filtering. The fusion processing includes two steps. The first step is to estimate the optimal position of an observed robot M^*_{Rj} by minimizing a weighted least-square criterion:

$$J(M) = (M_{Rj} - M)^T C_{Rj}^{-1} (M_{Rj} - M) + (M_{Rij} - M)^T C_{Rij}^{-1} (M_{Rij} - M). \quad (9)$$

More specifically, the first step of parallel fusion algorithm is to find the optimal position that minimizes the performance criterion (9) such that

$$M^*_{Rj} = \arg\min_M J(M) \quad (10)$$

By taking the derivative of (9) with respect to M, the necessary condition of local optimal solution of (10) can be obtained as follows

$$(M_{Rj} - M)^T C_{Rj}^{-1} + (M_{Rij} - M)^T C_{Rij}^{-1} = 0. \quad (11)$$

From (11), the local optimal solution of (10) is given by

$$\begin{aligned} M^*_{Rj} &= (C_{Rj}^{-1} + C_{Rij}^{-1})^{-1} C_{Rj}^{-1} M_{Rj} + (C_{Rj}^{-1} + C_{Rij}^{-1})^{-1} C_{Rij}^{-1} M_{Rij} \\ &= M_{Rj} + (C_{Rj}^{-1} + C_{Rij}^{-1})^{-1} C_{Rij}^{-1} (M_{Rij} - M_{Rj}) \\ &= M_{Rij} + (C_{Rj}^{-1} + C_{Rij}^{-1})^{-1} C_{Rj}^{-1} (M_{Rj} - M_{Rij}) \end{aligned} \quad (12)$$

Let $K_{ij} = (C_{Rj}^{-1} + C_{Rij}^{-1})^{-1} C_{Rij}^{-1}$ and $K_j = (C_{Rj}^{-1} + C_{Rij}^{-1})^{-1} C_{Rj}^{-1}$ denote Kalman gain matrices, expression (12) is simplified such that

$$M_{Rj}^{*} = M_{Rj} + K_{ij}(M_{Rij} - M_{Rj}) = M_{Rij} + K_{j}(M_{Rj} - M_{Rij}). \tag{13}$$

Therefore, expression (13) provides the optimal fusion result for the positional estimation of parallel fusion algorithm based on the performance criterion (9).
The second step of parallel fusion is to minimize the covariance matrix of the updated position of the observed robot:

$$C_{Rj}^{*} = E[(\overline{M}_{Rj} - M_{Rj}^{*})(\overline{M}_{Rj} - M_{Rj}^{*})^{T}]. \tag{14}$$

Substituting (13) into (14), we can obtain the following two results

$$\begin{aligned} C_{Rj}^{*} &= E\{[(\overline{M}_{Rj} - M_{Rj}) - K_{ij}(M_{Rij} - M_{Rj})][(\overline{M}_{Rj} - M_{Rj}) - K_{ij}(M_{Rij} - M_{Rj})]^{T}\} \\ &= C_{Rj} - C_{Rj}K_{ij}^{T} - K_{ij}C_{Rj} + K_{ij}C_{Rij}K_{ij}^{T} + K_{ij}C_{Rj}K_{ij}^{T}, \end{aligned} \tag{15}$$

$$\begin{aligned} C_{Rj}^{*} &= E\{[(\overline{M}_{Rj} - M_{Rij}) - K_{j}(M_{Rj} - M_{Rij})][(\overline{M}_{Rj} - M_{Rij}) - K_{j}(M_{Rj} - M_{Rij})]^{T}\} \\ &= C_{Rij} - C_{Rij}K_{j}^{T} - K_{j}C_{Rij} + K_{j}C_{Rj}K_{j}^{T} + K_{j}C_{Rij}K_{j}^{T}. \end{aligned} \tag{16}$$

From (15), the minimum covariance matrix can be obtained by taking the derivative of (15) with respect to K_{ij} and setting to zero such that

$$\frac{dC_{Rj}^{*}}{dK_{ij}} = -2C_{Rj} + 2K_{ij}(C_{Rj} + C_{Rij}) = 0. \tag{17}$$

Solving (17) for K_{ij} gives

$$K_{ij} = C_{Rj}(C_{Rj} + C_{Rij})^{-1} = (C_{Rj}^{-1} + C_{Rij}^{-1})^{-1}C_{Rij}^{-1}. \tag{18}$$

Expression (18) is the Kalman gain equation. Substituting (18) into (15), the minimum covariance matrix becomes

$$C_{Rj}^{*} = C_{Rj} - C_{Rj}K_{ij}^{T} - K_{ij}C_{Rj} + C_{Rj}K_{ij}^{T} = C_{Rj} - K_{ij}C_{Rj} = (I - K_{ij})C_{Rj}. \tag{19}$$

Using the same procedure discussed above, similar result also can be obtained from (16) such that

$$K_{j} = C_{Rij}(C_{Rj} + C_{Rij})^{-1} = (C_{Rj}^{-1} + C_{Rij}^{-1})^{-1}C_{Rj}^{-1}, \tag{20}$$

$$C_{Rj}^{*} = C_{Rij} - K_{j}C_{Rij} = (I - K_{j})C_{Rij}. \tag{21}$$

Based on equations (13) and (18)-(21), the parallel fusion algorithm to estimate the optimal position and minimum positional uncertainty of an observed robot in world coordinate system is summarized as follows:

Kalman Gain Matrix:

$$K_{ij} = C_{Rj}(C_{Rj} + C_{Rij})^{-1} = (C_{Rj}^{-1} + C_{Rij}^{-1})^{-1} C_{Rij}^{-1}, \tag{22}$$

Updated Position:

$$M_{Rj}^{*} = M_{Rj} + K_{ij}(M_{Rij} - M_{Rj}), \tag{23}$$

Updated Covariance Matrix:

$$C_{Rj}^{*} = C_{Rj} - K_{ij} C_{Rj} = (I - K_{ij}) C_{Rj}. \tag{24}$$

Or,
Kalman Gain Matrix:

$$K_{j} = C_{Rij}(C_{Rj} + C_{Rij})^{-1} = (C_{Rj}^{-1} + C_{Rij}^{-1})^{-1} C_{Rj}^{-1}, \tag{25}$$

Updated Position:

$$M_{Rj}^{*} = M_{Rij} + K_{j}(M_{Rj} - M_{Rij}), \tag{26}$$

Updated Covariance Matrix:

$$C_{Rj}^{*} = C_{Rij} - K_{j} C_{Rij} = (I - K_{j}) C_{Rij}. \tag{27}$$

The symbol " $\circ$ " is employed to represent the parallel fusion operation. For instance, we can simplify the presentation of parallel fusion equations (22)-(24) as:

$$M_{Rj}^{*} = M_{Rj} \circ M_{Rij}, \tag{28}$$

$$C_{Rj}^{*} = C_{Rj} \circ C_{Rij} \tag{29}$$

Similarly, the presentation of parallel fusion equations (25)-(27) can be simplified as:

$$M_{Rj}^{*} = M_{Rij} \circ M_{Rj}, \tag{30}$$

$$C_{Rj}^{*} = C_{Rij} \circ C_{Rj} \tag{31}$$

Compare equations (28)-(29) with (30)-(31), it is clear that the parallel fusion operation has *commutative* property. Moreover, because of $I - K_{ij} = C_{Rij}(C_{Rj} + C_{Rij})^{-1}$ and

$I - K_j = C_{Rj}(C_{Rj} + C_{Rij})^{-1}$, the updated covariance matrix becomes

$$C^*_{Rj} = C_{Rij}(C_{Rj} + C_{Rij})^{-1} C_{Rj} = C_{Rj}(C_{Rj} + C_{Rij})^{-1} C_{Rij}\,. \tag{32}$$

Expression (32) leads to the fact that the matrix norm of updated covariance matrix, $\left\| C^*_{Rj} \right\|$, will not be larger than that of covariance matrices C_{Rj} and C_{Rij} (Golub & Van Loan, 1996). More specifically, expression (32) leads to the following result

$$\left\| C^*_{Rj} \right\| \le \min(\left\| C_{Rj} \right\|, \left\| C_{Rij} \right\|)\,. \tag{33}$$

Therefore, the parallel fusion operation also has *associative* property, which guarantees that the final fusion result has minimum matrix norm irrelevant to the order of fusion operation. These two properties are helpful to generalize the parallel fusion algorithm into the sensor fusion procedure of multiple robots. Suppose that robot j is observed by robot i at instant k, where $1 \le i, j \le n$, and $\mathrm{i} \ne \mathrm{j}$, we can write the parallel fusion formula such that:

$$(M_{Rj})_{k+1} = (M_{Rj} \circ M_{R1j} \circ M_{R2j} \circ \ldots \circ M_{Rnj})_k\,, \tag{34}$$

$$(C_{Rj})_{k+1} = (C_{Rj} \circ \mathrm{C}_{R1j} \circ \mathrm{C}_{R2j} \circ \ldots \circ \mathrm{C}_{Rnj})_k\,, \tag{35}$$

where $(M_{Rj})_{k+1}$ and $(C_{Rj})_{k+1}$ represent the updated global position and the corresponding covariance matrix of robot j at next instant $k+1$, respectively. In equations (34) and (35), any two terms of $(M_{Rij}, C_{Rij})_k$, $1 \le i, j \le n$, can be chosen to perform the parallel fusion. The output of the first fusion is used as the input of next step, and then the optimal position of robot j with minimum covariance matrix can be obtained after several iterations.

Figure 5 depicts the complete steps of parallel fusion. As the case shown in Fig. 5(a), there are three sets of parameters of serial fusion obtained from other robots, which observe robot 2. Using the proposed parallel fusion scheme, the parameters of serial fusion $(M_{R12}, C_{R12})_k$, $(M_{R32}, C_{R32})_k$, $(M_{R42}, C_{R42})_k$ and $(M_{R2}, C_{R2})_k$ can be merged in three subsequent steps as depicted in Fig. 5(b). A new positional parameters of robot 2 at instant $k+1$, $(M_{R2}, C_{R2})_{k+1}$, is obtained and updated accordingly. We write parallel fusion formula following the representation in Fig. 5(b):

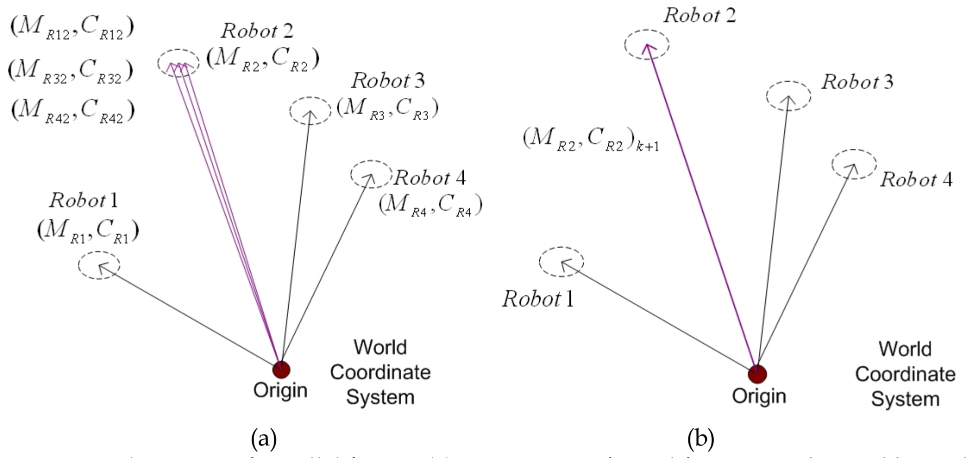


Fig. 5. Complete steps of parallel fusion. (a) Parameters of serial fusion are obtained by each robot which observes robot 2. (b) Parameters are fused in three subsequent steps using parallel fusion.

$$(M_{R2})_{k+1} = (M_{R12} \circ M_{R32} \circ M_{R42} \circ M_{R2})_{\mathrm{k}} \ , \tag{36}$$

$$(C_{R2})_{k+1} = (C_{R12} \circ \mathrm{C}_{R32} \circ \mathrm{C}_{R42} \circ \mathrm{C}_{R2})_{\mathrm{k}} \ . \tag{37}$$

We can also rewrite (36) and (37) to another form using the commutative and associative properties:

$$(M_{R2})_{k+1} = (M_{R42} \circ M_{R32} \circ M_{R12} \circ M_{R2})_{\mathrm{k}} \tag{38}$$

$$(C_{R2})_{k+1} = (\mathrm{C}_{R42} \circ \mathrm{C}_{R32} \circ \mathrm{C}_{R12} \circ C_{R2})_{\mathrm{k}} \tag{39}$$

In the following section, the performance of the proposed serial and parallel fusion algorithms will be validated by practical experiments.

5. Experimental Results

The developed system has been implemented on two experimental mobile robots H1 and H2 developed in our lab. Each robot has an on-board industrial PC (IPC), which is connected to the internet via wireless LAN. On top of the robot, a stereo vision module was installed in order to estimate the position of the observed robot and build the local map of the experimental environment. The robots were equipped with a motor control card for driving two independent wheels. Fig. 6 shows the robots H1 and H2 in the experiments. Figs. 7(a) and (b) show the experimental environment in the lab.
Two experiments were conducted to verify the effectiveness of the proposed method of cooperation localization. One experiment used a single robot H1 to build a map. The second experiment used two robots, H1 and H2, to cooperatively build an environmental map with multi-robot localization. In these two experiments, the robots are set to only have four

orientation angles: 0°, 90°, 180°, and 270°. Therefore, the orientation of the robot can be supposed to be known without uncertainty. This assumption leads a simplified implementation of the proposed algorithm, which does not consider the information of robot orientation. For instance, expressions (7) and (8) can be simplified such that

$$M_{Rij} = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f(M_{Ri}, M_{Eij}) \\ g(M_{Ri}, M_{Eij}) \end{bmatrix} = M_{Ri} + \begin{bmatrix} \cos\theta_r & -\sin\theta_r \\ \sin\theta_r & \cos\theta_r \end{bmatrix} M_{Eij}, \tag{40}$$

$$C_{Rij} = J \begin{bmatrix} C_{Ri} & \mathbf{0}_{2\times 2} \\ \mathbf{0}_{2\times 2} & C_{Eij} \end{bmatrix} J^T, \tag{41}$$

with

$$M_{Ri} = \begin{bmatrix} x_r \\ y_r \end{bmatrix},\ M_{Eij} = \begin{bmatrix} x_e \\ y_e \end{bmatrix}, \text{and } J = \begin{bmatrix} 1 & 0 & \cos\theta_r & -\sin\theta_r \\ 0 & 1 & \sin\theta_r & \cos\theta_r \end{bmatrix}.$$

However, the performance and convergence of the proposed algorithm will not be influenced. In the following, these two experimental results are compared and discussed.

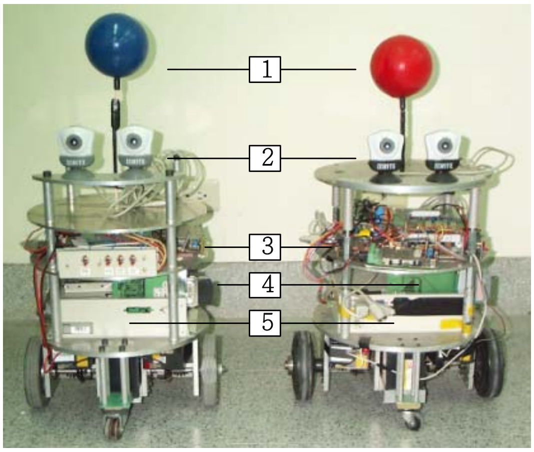


Fig. 6. Front view of robots H1 and H2. H1 on the left is with a blue ball and H2 with a red ball. Both robots are equipped with (1) a blue or red color ball, (2) stereo vision module with two CMOS cameras, (3) motion control module, (4) wireless LAN card, (5) on-board IPC.

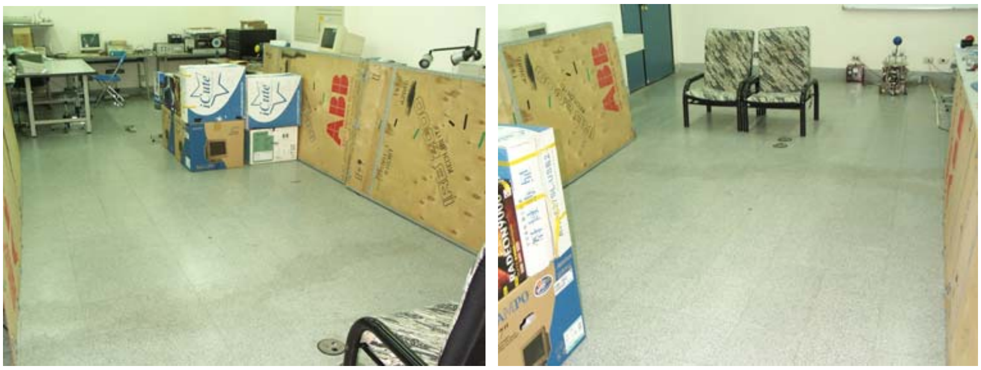

(a) (b)

Fig. 7. Experimental environment.

***Remark* 2**: The covariance matrices C_{Ri} and C_{Eij} in (41) are calibrated by practical testing for each robot. In the experiments, the covariance matrices for each robot are calibrated by

$$\text{Robot H1: } C_{R1}^{(k+1)} = C_{R1}^{(k)} + \begin{bmatrix} 0.0596 & 0 \\ 0 & 0.0248 \end{bmatrix} \text{ and}$$

$$C_{E12} = \begin{bmatrix} 0.0430 & 0 \\ 0 & 0.0532 \end{bmatrix},$$

$$\text{Robot H2: } C_{R2}^{(k+1)} = C_{R2}^{(k)} + \begin{bmatrix} 0.0478 & 0 \\ 0 & 0.0513 \end{bmatrix} \text{ and}$$

$$C_{E21} = \begin{bmatrix} 0.0595 & 0 \\ 0 & 0.0793 \end{bmatrix},$$

where $C_{Ri}^{(k)}$ denotes the positional covariance matrix of robot R_i at time step k and will be accumulated step-by-step when the robot travels due to the odometry measurement noise.

Experiment 1: Single robot localization and map-building

In this part, robot H1 was commanded to build a map of the environment alone. The localization of the robot is using the information from odometry only such that

$$x_r^{(k+1)} = x_r^{(k)} + v_r^{(k)} \cos\theta_r \text{, and } y_r^{(k+1)} = y_r^{(k)} + v_r^{(k)} \sin\theta_r \text{,} \tag{42}$$

where $v_r^{(k)}$ is the linear velocity of the robot at time step k. The linear velocity is calculated by counting encoder signals from two driving wheels. Because of $\theta_r = \{0^\circ, 90^\circ, 180^\circ, 270^\circ\}$, the angular velocity of the robot can be ignored in the experiments.

Fig. 8 shows the experimental results. In Fig. 8, the dotted lines represent actual environmental shape. Solid lines show the map established by the robot H1. Six positions A, B, C, D, E and F were recorded as the robot moved around the environment. Table 2 shows the position parameters of H1 of these points. In Table 2, it is observed that as H1 moves through point A to F subsequently, the standard deviations of x and y coordinates become larger and larger. This means that H1 is getting lost of its position estimation because the accumulated error from dead reckoning. From Table 2, we also see that the standard deviations of from A to F are increasing. This implies the error between the actual and estimated positions becomes larger as the robot travels further.

Experiment 2: Cooperative visual localization and map-building

In this part, robots H1 and H2 worked together to cooperatively localize each other and built a map of the environment. Fig. 9 shows the experimental results. In Fig. 9(a), seven positions of H1 were recorded. In Fig. 9(b), five positions of H2 were recorded. The black points are the recorded trajectory of H1 and H2, respectively. The dotted lines are actual environmental shape. Solid lines represent the maps established by H1 and H2 respectively. In this experiment, H1 was observed by H2 at point D of H1. The robot H2 was observed by H1 at points C and E of H2.

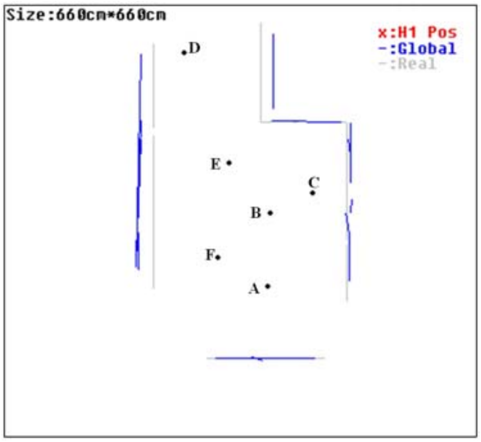


Fig. 8. Map-building result of single robot H1.

The position estimation results of H1 and H2 in experiment 2 are shown in Table 3 and 4, respectively. From Table 3, we see that when H1 moved through point A, B and C subsequently, the standard deviations of x and y coordinates increase. However, when H1 moved to point D and was observed by H2, parallel fusion of H1 occurred. So the position uncertainty of robot H1 and the error between actual and estimated position became smaller as expected. The similar results also can be confirmed for robot H2. As shown in Table 4, since H2 was observed by H1 at points C and E, the position uncertainty and the positional error were reduced at point C and E as expected.

	Actual Position(cm)	Estimated Position(cm)	Standard Deviation	Move Counts	Move Distance(cm)
A	(37,-98)	(40,-100)	σ_x=4.001 σ_y=3.600	1	40
B	(38,4)	(43.49,9.94)	σ_x=8.169 σ_y=8.229	11	270
C	(95,28)	(104.5,37.83)	σ_x=12.348 σ_y=11.356	22	540
D	(-60,240)	(-75.72,253.63)	σ_x=14.885 σ_y=13.943	33	930
E	(0,60)	(-14.09,83.92)	σ_x=16.808 σ_y=15.540	41	1170
F	(-17,-47)	(-32.36,-64.83)	σ_x=18.148 σ_y=16.084	46	1320

Table 2. Position parameters of H1 in experiment1.

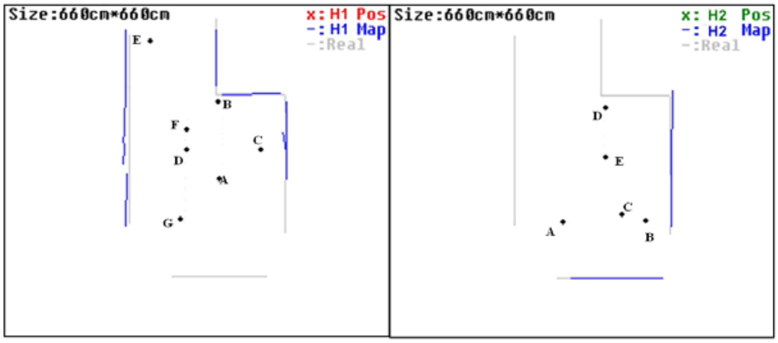


Fig. 9. Map-building results in experiment 2: (a) by H1 (b) by H2.

	Actual Position(cm)	Estimated Position(cm)	Standard Deviation	Move Counts	Move Distance(cm)	Observed by H2
A	(40,-10)	(40.01,-10)	σ_x=3.904 σ_y=3.610	1	10	No
B	(33,115)	(40.34,120.1)	σ_x=11.023 σ_y=8.854	16	280	No
C	(100,31)	(110.5,40.3)	σ_x=13.835 σ_y=12.109	26	430	No
D	(-14,35)	(-11.19,38.64)	σ_x=3.710 σ_y=3.620	28	530	Yes
E	(-76,225)	(-82.41,231.19)	σ_x=8.108 σ_y=7.554	41	740	No
F	(-5,70)	(-11.81,64.83)	σ_x=12.158 σ_y=11.034	52	910	No
G	(-12,-70)	(-21.31,-78.29)	σ_x=14.368 σ_y=12.947	60	1090	No

Table 3. Position parameters of H1 in experiment 2.

	Actual Position(cm)	Estimated Position(cm)	Standard Deviation	Move Counts	Move Distance(cm)	Observed by H1
A	(-28,-100)	(-30,-100)	σ_x=5.385 σ_y=6.196	3	30	No
B	(103,-91)	(110.49,-100)	σ_x=8.462 σ_y=9.306	10	190	No
C	(58,-92)	(60.2,-95.18)	σ_x=3.878 σ_y=4.356	13	240	Yes
D	(34,94)	(41.2,108.18)	σ_x=11.885 σ_y=12.043	28	570	No
E	(38,40)	(41.09,43.92)	σ_x=5.180 σ_y=5.354	34	740	Yes

Table 4. Position parameters of H2 in experiment 2.

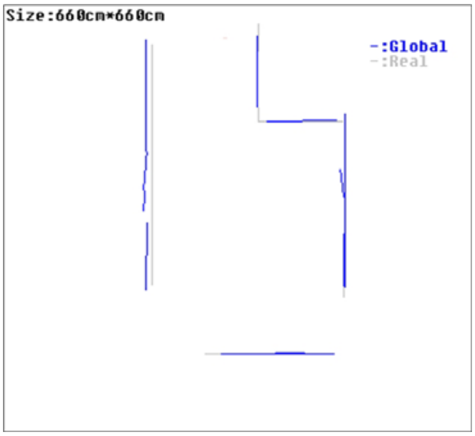


Fig. 10. Cooperative map-building result of H1 and H2.

Therefore, the experimental results validate that the proposed cooperative localization algorithm effectively reduces the position error of each robot in the cooperative robotic system.
Fig. 10 shows the integrated result of Fig. 9 constructed in the robot server through cooperation of robots H1 and H2. In Fig. 10, the gray lines represent actual environmental shape. Solid lines represent the global map established by robot H1 and H2 collaboratively. Because the position error of each robot was reduced by the cooperative localization, the error of the map also became smaller compared with the case in Fig. 8 without sensor fusion.

6. Conclusions

In this chapter, we presented a multi-robot cooperative localization scheme which does not restrict the number of robots in the system and guarantees the optimal fusion results with minimum covariance matrix. In this scheme, each robot is equipped with a stereo vision system to recognize the individual robot and find important features in environment. This is advantageous compared to a robotic system with sonar or laser sensors, where the robots have practical limitations in recognizing more than two robot-teammates. From practical experiments, we observe that parallel fusion can effectively reduce robot positional uncertainty as expected. Using serial and parallel fusion, one can increase the accuracy of robot localization and reduce the errors in map-building. In the future, we will first extend the experiments to a more complex environment with more robots. Theoretical improvements will also be investigated, such as the information transmission issue between each robot teammate, and how often the robots need to observe each other.

7. Acknowledgnent

This work was partly supported by the National Science Council of Taiwan, ROC under grant NSC 96-2628-E-009-162 and NSC 94-2218-E-009-008.

8. References

Fox, D.; Burgard, W.; Thrun, S. & Cremers, A.B. (1998). Position estimation for mobile robots in dynamic environments. *Proceedings of Fifteenth National Conference on Artificial Intelligence*, Madison, Wisconsin, pp. 983-988

Fox, D.; Burgard, W.; Kruppa, H. & Thrun, S. (1999). Collaborative multi-robot localization. *Proceedings of German Conference on Artificial Intelligence*, Bonn, Germany, pp. 255-266

Gamini Dissanayake, M. W. M.; Newman, P.; Clark, S.; Durrant-Whyte, H. F. & Csorba, M. (2001). A solution to the simultaneous localization and map-building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, Vol. 17, No. , pp. 229-241

Golub, G. H. & Van Loan, C. F. (1996). *Matrix computations – third edition*, Baltimore: The Johns Hopkins University Press

Jennings, C.; Murray, D. & Little, J. J. (1999). Cooperative robot localization with vision-based mapping. *Proceedings of IEEE International Conference on Robotics and Automation*, Detroit, USA, pp. 2659-2665

Nieto, J.; Guivant, J.; Nebot, E. & Thrun, S. (2002). FastSLAM: A factored solution to the simultaneous localization and mapping problem. *Proceedings of Eighteenth National Conference on Artificial Intelligence*, Alberta, Canada, pp. 593-598

Nieto, J.; Guivant, J.; Nebot, E. & Thrun, S. (2003). Real time data association for FastSLAM. *Proceedings of IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, pp. 412-418

Parker, L. E. (1998). ALLIANCE: An architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 2, pp. 220-240

Rekleitis, I. M.; Dudek, G. & Milios, E. (2003). Probabilistic cooperative localization and mapping in practice. *Proceedings of IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, pp. 1907-1912

Roumeliotis, S. I. & Bekey, G. A. (2002). Distributed multi-robot localization. *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 5, pp. 781-795

Smith, R. C. & Cheeseman, P. (1986). On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, Vol. 5, No. 4, pp. 56-68

Stroupe, A. W. & Balch, T. (2002). Collaborative probabilistic constraint-based landmark localization. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems,* Switzerland, pp. 447-453

Stroupe, A. W. & Martin, C. (2001). Distributed sensor fusion for object position estimation by multi-robot systems. *Proceedings of IEEE International Conference on Robotics and Automation*, Seoul, Korea, pp. 1092-1098

Thrun, S.; Fox, D.; Burgard, W. & Dellaert, F. (2001). Robust monte carlo localization for mobile robots, *Artificial Intelligence*, Vol. 128, No. 1-2, pp. 99-141

Weigel, T.; Gutmann, J.-S.; Dietl, M.; Kleiner A. & Nebel, B. (2002). CS Freiguring: Coordinating robots for successful soccer playing. *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 5, pp. 685-699

Zhang, L. & Ghosh, B. K. (2000). Line segment based map building and localization using 2D laser rangefinder. *Proceedings of IEEE International Conference on Robotics and Automation*, San Francisco, USA, pp. 2538-2543

Filtering Algorithm for Reliable Localization of Mobile Robot in Multi-Sensor Environment

Yong-Shik Kim[1], Jae Hoon Lee[1], Bong Keun Kim[1], Hyun Min Do[1] and Akihisa Ohya[2]
[1]UFRG, AIST, Central 2, 1-1-1 Umezono, Tsukuba, 305-8568, Japan
Email: yongshik.kim2009@gmail.com, {jh.lee, bk.kim, hyunmin.do}@aist.go.jp
[2]Intelligent Robot Laboratory, University of Tsukuba, Tsukuba, Japan
Email: ohya@roboken.esys.tsukuba.ac.jp

Abstract:
Localization problem of mobile robot in an environment of interest is one of research fields that are not still completely solved. Many methods to settle this problem are unceasingly being examined. Although a lot of researchers have dedicated themselves to this problem, it demands new but more accurate method by which mobile robot itself recognizes its own position while moving. It tries to find solution by making mention of various localization problems with estimation method in this chapter. Robot researchers had to recognize the current position information for movement control of mobile robot to last destination. It is being adopted from the conventional Kalman filter, which was proposed to reduce stochastic noise error, to complicated algorithms in the field of target tracking. The localization of the mobile robot occurred in various environments such as indoor, outdoor, warehouse, harbors, airports, and offices is based on estimation theories of target tracking field. The estimation method to be proposed in this chapter is mentioned with localization problem in the text. As the most widely used method for providing the current position of mobile robot, odometry is inexpensive method to implement in real time, but has vital demerit of accumulation of position error for long distance navigation. Various external sensors were used to supplement this problem, but the adopted sensors have also intrinsic errors. GPS, used as a global sensor in the outdoors, has a fatal weak point suffering from multi-path effects in the surrounding of buildings or trees. In this chapter, method for decreasing the above error occurred in localization problem are provided and embodied. This error is assumed as bias error. New nonlinear estimation method included in data integration method to reduce that error is derived and evaluated through experiment. The proposed integration method is provided to take full advantage of characteristics of mobile and outdoor environment sensors.

1. Introduction

Localization of moving object is one of the main subjects in the area of robotics. In particular, the service robot has aimed at providing position information for the running of the robot to support man's vital function. In the viewpoint of the robot, it is necessary to recognize the position in the given environment to carry out the desired duty in daily life. The robot may receive information from not only the sensors that robot has but also devices distributed on the ambient environment. There exist various sensing technologies to recognize the current position of mobile robot in the given environment. For such case, sensors can be classified as the following two categories in the view of information provider: i) Odometry, inertial measurement unit (inertial navigation system), ultrasonic sensor, laser range finder, moving camera such as stereo camera; ii) Camera array system, radio frequency identification (RFID), global positioning system (GPS), starLITE sensor suite, ultrasonic satellite (U-SAT), stargazer, laser range finder with reflectors. Odometry is the most widely used method for determining the current position of the mobile robot. An odometric sensor is simple, inexpensive, and easy to implement in real time. The disadvantage of odometry, however, is the accumulation of position errors for long distance navigation. In order to solve this problem, absolute and global information to the running robot is needed. As one of the aiding methods for that information of a mobile robot, the widely used sensors include starLITE sensor suite, vision system, GPS, and RFID. Differential GPS (DGPS) method has been developed to reduce the odometry error in real time. Nevertheless, the DGPS accuracy cannot be guaranteed all the time in environments where partial satellite occlusion and multi-path effects between buildings can prevent normal GPS receiver operation. Therefore, for the localization of mobile robot, it is indispensable to consider each characteristic of the used sensors because each sensor receives data with different method. It can be considered as registration error of mobile sensors. Data registration problem can be settled down by pre-processing of sensor data [9], [21]. In this chapter, sensor data transformed from local coordinate reference system to global coordinate reference system is used as inputs of integration filter.

Information fusion of various sensors enables one to obtain better information than independent that of each sensor. Sensor data fusion is indispensable for localization of a mobile robot in the given environment. Data fusion techniques are used to employ a number of sensors and to fuse the information from all of these sensors and environment. But, sensor data or information may be of different types. In that case, integration can be used as a special form of data fusion after sensor registration. Various integration methods using dead-reckoning and external information have been in the literature [1], [2], [5], [11], [14], [15], [18]-[20]. Nebot and Durrant-Whyte [16] presented the design of a high-integrity navigation system for use in large autonomous mobile vehicles. Decentralized integration architecture was also presented for the fusion of information from different asynchronous sources. The integration includes a complementary fusion [2], [8], a centralized integration [2], [13], and a distributed integration method [4], [6], [7], [22]. For integrating information of DGPS and odometric data, a complementary integration approach is proposed [2], [3], [19]. The used integration filtering method uses odometry data as system state and DGPS data as measurements. It needs difference between two sensor data as input variables of filter. The extended Kalman filter (EKF) [2] is used as integration filter to estimate sensor data error. In addition, the filtering output is resent to odometry system to correct robot position. However, even after data integration, the undesirable result may be obtained. This is due to multi-path phenomenon of the DGPS sensor [8], [16], [19]. Novoselov et al. [17] presented the algorithm

based on the Schmidt-Kalman filter for mitigating the effects of residual biases on sensor attitude error and measurement offset and scale errors. Huang and Tan [12] investigated the characteristics of DGPS measurements under urban environments. In addition they proposed novel DGPS noise processing techniques to reduce the chances of exposing the EKF to undesirable DGPS measurements due to common DGPS problems such as blockage and multipath. When one of several sensors provides bad information in multi-sensor structure, the proposed mechanism detects a sensor fault from integrating result and compensates information by bias estimation. The used bias estimation was originated by Friedland [10]. In [10], the estimation mechanism is composed of two parts: bias-free filter and bias filter. The estimation of the bias is decoupled from the computation of the bias-free estimate of the state. Therefore, the aim of this study is to use each characteristic of sensor data and to develop an integration structure dealing with those data for the localization of mobile robot in spite of sensor data fault or bias error.
This paper is organized as follows. In Section 2, for localization of mobile robot two different integration sensors and their characteristics are described. The problem is formulated to integrate information from odometry sensor and ambient environment, and integration structure is proposed in Section 3. It is also proposed an integration method with bias estimation to compensate bias error. The problem with data fault phenomena is mentioned and recovered. Outdoor robot Yamabico is used to verify the proposed mechanism in Section 4. Section 5 concludes the paper.

2. Localization without Bias Compensation

For the localization of mobile robot using odometry and DGPS sensors, an integration mechanism taking use of characteristics of sensor data is needed. This section will, therefore, focus on the integration method of two sensors. It is worthwhile to note that integrated localization depends on characteristics of sensor data. So, the integration problem can be stated as how to best extract useful information from multiple sets of data with different characteristics being available.

2.1. Characteristics of Sensors

First, odometry system equation as dead-reckoning and DGPS for the localization aid are briefly discussed, respectively. Odometric sensor is a positioning sensor which estimates both position and orientation of the mobile robot by integrating the measurement of driving wheel rotations. The robot's position is defined as $x(k) = [\eta(k) \;\; \xi(k) \;\; \theta(k)]^T$ and its error covariance is denoted as $\sum_{x(k)}$. Then, the robot position and its estimated error are represented as follows:

$$x(k+\Delta T) = \begin{bmatrix} \eta(k) \\ \xi(k) \\ \theta(k) \end{bmatrix} + \begin{pmatrix} \Delta T V(k)\cos(\theta(k)) \\ \Delta T V(k)\sin(\theta(k)) \\ \Delta T \phi(k) \end{pmatrix} + w, \tag{1}$$

$$\sum\nolimits_{x(k+\Delta T)} = J \sum\nolimits_{x(k)} J^T + K(k) \sum\nolimits_{V(k)} K(k)^T + \sum\nolimits_{N}, \tag{2}$$

$$\sum\nolimits_{x(k)} = \begin{pmatrix} \sigma_{\eta_0}(k)^2 & \sigma_{\eta_0\xi_0}(k) & \sigma_{\eta_0\theta_0}(k) \\ \sigma_{\xi_0\eta_0}(k) & \sigma_{\xi_0}(k)^2 & \sigma_{\xi_0\theta_0}(k) \\ \sigma_{\theta_0\eta_0}(k) & \sigma_{\theta_0\xi_0}(k) & \sigma_{\theta_0}(k)^2 \end{pmatrix} \tag{3}$$

where ΔT is a sampling period. $V(k)$, $\theta(k)$, and $\phi(k)$ is velocity, orientation, and angular velocity, respectively. $J(k)$ is Jacobian of $x(k)$ with respect to η, ξ, and θ. $K(k)$ is Jacobian of $x(k)$ with respect to V and θ.
In this work, DGPS (Trimble DSM212L) receiver is used as external sensor. DGPS can reduce the measurement error within one or several meters from original GPS data. The output data format is NMEA-0183 which offers a series of characters through a RS232C communication channel. Accuracy and resolution of the DGPS receiver was tested in the wide parking lot with the RTK-GPS. The DGPS sensor used in this study has data error of 30cm-50cm. Fig. 1 shows the DGPS sensor using in this experiment.

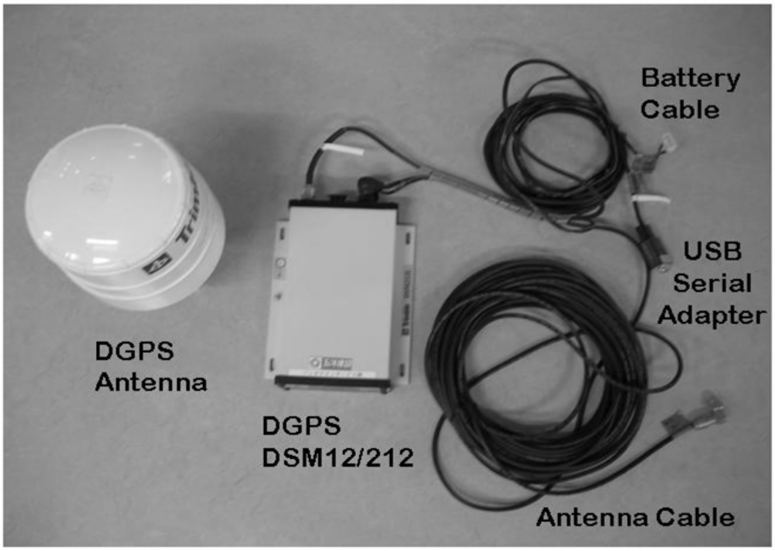


Fig. 1. DGPS experiment equipment.

2.2. Stochastic Problem Formulation

Considering such a kinematics of odometry, the following nonlinear dynamic system and measurement equations can be written:

$$x(k) = f[x(k-1)] + \omega(k-1),\ k = 1,2,\cdots, \tag{4}$$

$$z_i(k) = h_i[x(k)] + \upsilon_i(k), \quad i = 1,\cdots,N \tag{5}$$

where $x(k) \in \Re^n$ is the state vector at time k, f is a nonlinear function, $\omega(k) \in \Re^n$ is the process noise, $z_i(k) \in \Re^{m_i}$ is the observation vector at *i*th local sensor, $h_i \in \Re^{m_i \times n}$ is the nonlinear measurement function, $\upsilon_i(k) \in \Re^{m_i}$ is the observation noise, $m_1 + \cdots + m_N = m$, and *N* is the number of sensors. In order to obtain the predicted state $\hat{x}(k \mid k-1)$, the nonlinear function in (4) is expanded in Taylor series around the latest estimate $\hat{x}(k-1 \mid k-1)$ with terms up to first order, to yield the first-order EKF. The vector Taylor series expansion of (4) up to first order is

$$x(k) = f[\hat{x}(k-1 \mid k-1)] + f_x(k-1)[x(k-1) - \hat{x}(k-1 \mid k-1)] + \mathrm{HOT} + \omega(k-1) \quad (6)$$

where HOT represents the higher-order terms and

$$f_x(k-1) = [\nabla_x f(x)']' \,|_{x=\hat{x}(k-1|k-1)} \quad (7)$$

is the Jacobian of the vector *f* evaluated with the latest estimate of the state.

2.3. Complementary Integration without Bias Compensation

For integrating information of DGPS and odometric data, a complementary integration approach is proposed [2], [3], [19]. The complementary configuration is shown in Fig. 2 [2]. According to the complementary integration scheme, the odometry sensor data is used as system information and DGPS data is used as measurements. It needs difference between two sensor data as input variables of filter. An EKF is used as integration filter to estimate sensor data error. In addition, the key in the suggested filter design is that filter output is resent to odometry system to correct robot position.

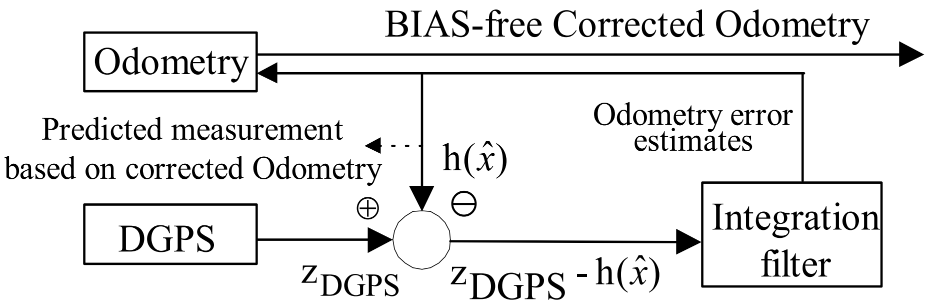

BIAS-free Corrected Odometry
Odometry
Predicted measurement based on corrected Odometry
h($\hat{x}$)
Odometry error estimates
⊕
⊖
DGPS
Integration filter
z_{DGPS}
z_{DGPS} - h($\hat{x}$)

Fig. 2. Configuration of complementary integration.

For the integration filtering, the covariance matrix and state estimate equations are as follows:

i) Time update (prediction)

$$\hat{x}(k \mid k-1) = f[\hat{x}(k-1 \mid k-1)],$$

$$P(k \mid k-1) = f_x(k-1)P(k-1 \mid k-1)f_x'(k-1) + Q(k-1). \quad (8)$$

ii) Measurement update

$$\hat{x}(k \mid k) = \hat{x}(k \mid k-1) + W(k)[z(k) - h_x(k)],$$

$$P(k \mid k) = P(k \mid k-1) - W(k)S(k)W'(k), \tag{9}$$

where $P(k|k)$ is the covariance matrix and $\hat{x}(k \mid k)$ is the state estimate vector. $h_x(k) = [\nabla_x h(x)']' |_{x=\hat{x}(k|k-1),j}$ is the Jacobian of the vector h evaluated at the predicted state $\hat{x}(k \mid k-1)$.

2.4. Data Fault by Multi-Path Phenomenon

DGPS sensor provides rather accurate information for long distance navigation as a sensing method providing an absolute position value. However, this accuracy cannot be guaranteed all the time in environments where partial satellite occlusion and multipath effects between buildings can prevent normal GPS receiver operation. Fig. 3 shows data fault by multi-path of RTK-GPS in the surrounding of buildings. Therefore, the correct position information for localization of mobile robot is not provided because of fault error by multipath of DGPS sensor,. In this paper, this fault error is considered as bias error of sensor.

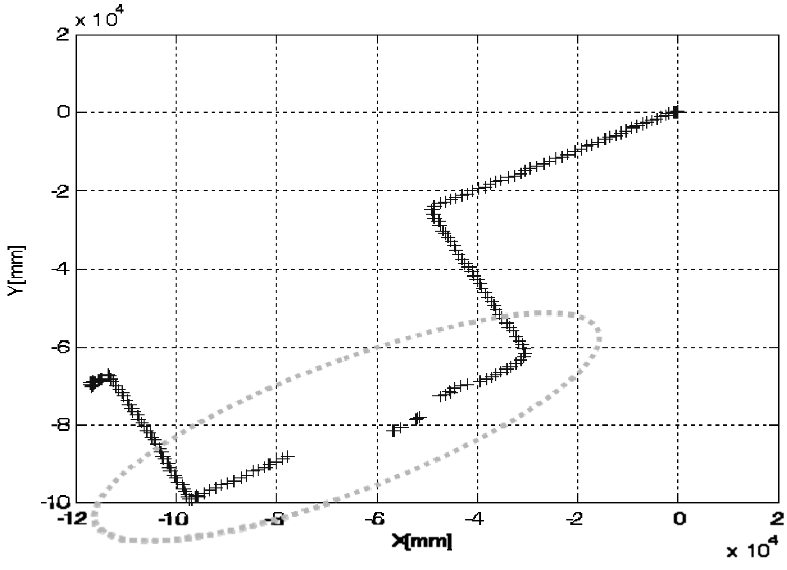


Fig. 3. Multi-path result of RTK-GPS data.

3. Integration Method with Bias Error Compensation

3.1. Problem Formulation

Assuming bias error in the system model and sensor model, the nonlinear dynamic system and measurement equations are as follows:

$$x(k) = f[x(k-1)] + B(k-1)b(k-1) + \omega(k-1),\ k = 1, \cdots, \tag{10}$$

$$z_i(k) = h_i[x(k)] + C_i(k)b(k) + \upsilon_i(k), \quad i = 1, \cdots, N, \tag{11}$$

$$b(k+1)=b(k) \tag{12}$$

where $x(k) \in \Re^n$ is the state vector at time *k*, *f* and h_i is a nonlinear functions, $\omega(k) \in \Re^n$ is the process noise, $z_i(k) \in \Re^{m_i}$ is the observation vector at *i*th local sensor, $\upsilon_i(k) \in \Re^{m_i}$ is the observation noise, and *N* is the number of sensors. $b(\cdot) \in \Re^d$ denote constant bias vectors and enter linearly. $B \in \Re^{n \times d}$ and $C_i \in \Re^{m_i \times d}$ denote how to bias vector enters into the dynamics and sensor model. In order to obtain the predicted state $\hat{x}(k \mid k-1)$, the nonlinear function in (10) is expanded in Taylor series around the latest estimate $\hat{x}(k-1 \mid k-1)$ with terms up to first order, to yield the first-order EKF. The vector Taylor series expansion of (10) up to first order is

$$\begin{aligned} x(k) = {} & f[\hat{x}(k-1 \mid k-1)] + f_x(k-1)[x(k-1) - \hat{x}(k-1 \mid k-1)] \\ & + \mathrm{HOT} + B(k-1)b(k-1) + \omega(k-1) \end{aligned} \tag{13}$$

where HOT represents the higher-order terms and

$$f_x(k-1) = [\nabla_x f(x)']' \big|_{x=\hat{x}(k-1|k-1)} \tag{14}$$

is the Jacobian of the vector f evaluated with the latest estimate of the state.

3.2 Estimation with Bias Compensation

In this paper, two-stage estimator by Friedland [10] is used. The estimation mechanism is composed of two parts: bias-free filter and bias filter. The estimation of the bias is decoupled from the computation of the bias-free estimate of the state.

A. Bias-free estimator

The estimation progress of bias-free filter is as follows:

1) Predict bias-free covariance matrix

$$P(k \mid k-1) = f_x(k-1)P(k-1 \mid k-1)f_x(k-1)' + Q(k-1)$$

2) Predict bias-free state estimate vector

$$\hat{x}(k \mid k-1) = f_x(k-1)\hat{x}(k-1 \mid k-1)$$

3) Predict bias-free measurement

$$\hat{z}(k \mid k-1) = h_x(k)\hat{x}(k \mid k-1)$$

4) Compute bias-free Kalman gain

$$K_x(k) = P(k \mid k-1)h_x'(k)(h_x(k)P(k \mid k-1)h_x'(k) + R(k))^{-1}$$

5) Update bias-free covariance matrix

$$P(k \mid k) = P(k \mid k-1) - K_x(k)S(k)K_x'(k)$$

where $S(k) = H(k)P(k \mid k-1)H'(k) + R(k)$ is covariance matrix of the innovation vector $\tilde{z}(k \mid k-1)$.

6) Receive measurement data

$$z(k)$$

7) Calculate bias-free innovation vector

$$\tilde{z}(k \mid k-1) = z(k) - \hat{z}(k \mid k-1)$$

8) Update bias-free estimate of state vector

$$\hat{x}(k \mid k) = \hat{x}(k \mid k-1) + K_x(k)\tilde{z}(k \mid k-1)$$

B. Bias estimator

The procedure for calculating estimate of bias filter in the presence of bias error is as follows:

1) Update Ux matrix

$$U_x(k) = F(k)V_x(k) + B(k)$$

2) Compute T matrix

$$T(k) = h_x(k)U_x(k) + C(k)$$

3) Compute Vx matrix using the bias-free Kalman gain Kx

$$V_x(k) = U_x(k) - K_x(k)T(k)$$

4) Compute bias covariance matrix

$$M(k) = M(k-1) - M(k-1)T'(k)(h_x(k)P_x(k \mid k-1) \times h_x'(k) + R(k) + T(k)M(k-1)T'(k))^{-1}T(k)M(k-1)$$

5) Compute bias Kalman gain

$$K_b(k) = M(k)(V_x'(k)h_x'(k) + C'(k))R^{-1}(k)$$

6) Compute bias estimate using the bias-free innovation vector

$$\hat{b}(k) = (I - K_b(k)T(k))\hat{b}(k-1) + K_b(k)\tilde{z}(k \mid k-1)$$

7) Compute bias correction

$$\sigma_k = V_x(k)\hat{b}(k)$$

8) Compute state estimate in the presence of bias error

$$\hat{x}_b(k \mid k) = \hat{x}(k \mid k) + \sigma(k)$$

C. Complementary Integration with Bias Compensation

First, the integration configuration proposed in this paper is shown in Fig. 4. According to the suggested integration scheme, the odometry sensor data is used as system information and DGPS data is used as measurements. It needs difference between two sensor data as input variables of filter. An EKF is used as integration filter to estimate sensor data error. In addition, the key in the suggested filter design is that filter output is resent to odometry system to correct robot position. In the suggested filter, contrary to the standard complementary integration, the integration filter is used for mitigating the effect of biases, since there is undesirable DGPS bias error in DGPS problems such as multipath phenomenon under urban environments. The proposed mechanism detects a data fault from the integrated result and compensates information by bias estimation.

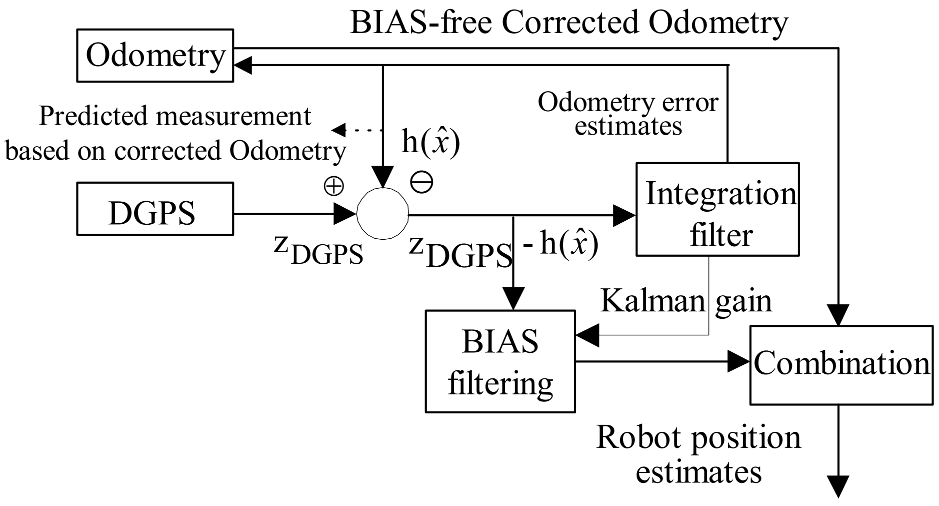


Fig. 4. Complementary integration with bias compensation.

4. Experiment and Results

4.1 System Configuration

In order to verify the integration method proposed in Section 3, position data of a Yamabico robot was received in outdoor environment. Fig. 5 depicts a configuration of the hardware system including three sensors and Yamabico robot for outdoor experiment. The used Yamabico robot was manufactured for outdoor experiment. The size of two wheels is bigger than those of indoor Yamabico robot. Air pressure of these wheels has to be checked before outdoor experiment.

For the experiment take into accounting characteristic of sensors, the parking lots and the surrounding of overcrowded buildings was selected as the experiment place. Data acquisition time of three different sensors equipped to the robot is different from each other. The sampling period of DGPS was 1sec. The sampling period of odometric sensor was about 5msec. Data from each sensor were received using a program considering these data receiving delay. Data communication between two sensors and laptop computer is transferred via RS232C.In this paper, an RTK-GPS as a measure for accuracy of DGPS was used. Data acquisition of the DGPS & Odometric receiver was carried out from the parking lots to the surrounding of building with the RTK-GPS.

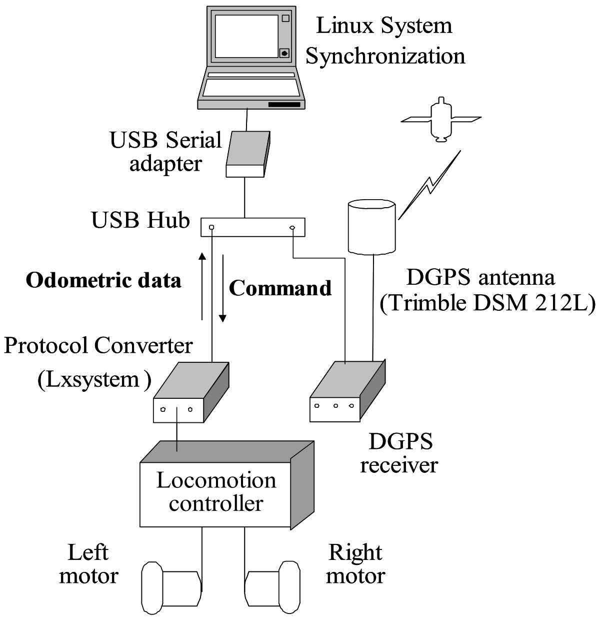


Fig. 5. H/W configuration.

4.2 Experiments and Simulation

A comparison of data received was carried out through computer simulation. Data registration was implemented using position data obtained from outdoor experiment. Basically, sensor data must transform to the global coordinate reference system. However, sensor data is not always transformed in all part. According to place or environment, a part of data can be lost. In such case, even if the coordinates is transformed, it can use no information in the interval where data was lost. Hence, in order to take full advantage of information, it is necessary to integrate considering data fault. In the parking lots without objects to its surrounding, the receiving condition of DGPS sensor is good. But, on the contrary, in the surrounding of the crowded buildings, DGPS data did not provide accurate position data due to the multi-path effect. Fig. 6 shows the result integrated by the conventional complementary method and the proposed complementary method. In spite of integrating two sensor data, in the surrounding of the building, the conventional complementary result did not provide accurate position information due to the multi-path effect. However, in the proposed method, robot position was corrected using data recovered by the bias estimation when the DGPS data can't be trusted. In order to detect bias error by the DGPS multi-path phenomena, the following normalized innovation square formula was used [2]

$$\tilde{r}'(k \mid k-1) S^{-1}(k) \tilde{r}(k \mid k-1)$$

where $\tilde{r}$ denotes the innovation vector and S denotes covariance matrix of the innovation vector. This value is compared with a threshold value determined by the chi-square distribution. In this study, for 95% probability, the theoretical threshold value is 7.81.

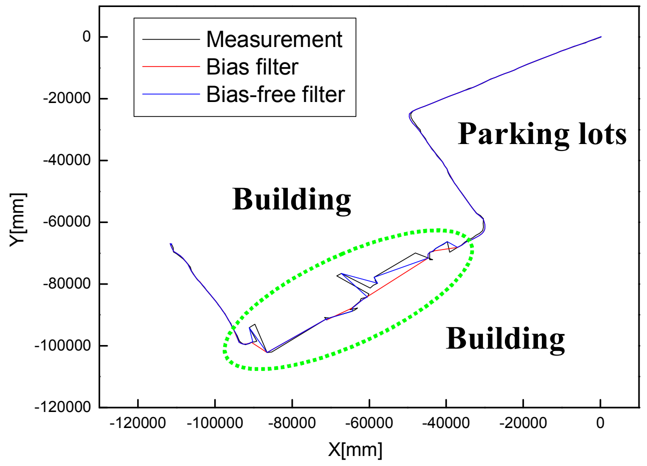


Fig. 6. Comparison of the standard and proposed complementary method.

5. Conclusions

In this paper, bias estimation based data integration method was provided. It is shown that the sensor data must be integrated with selection according to the sensing environment. Odometry and DGPS data can generally be used to localize mobile robot in outdoor environment. In the case there is slip phenomena by multi-path phenomena of DGPS sensor, however, the previous integration method can't solve data fault. The proposed data integration method is adequate to this condition. In this paper, the problem of data fault was solved by bias estimation. The proposed method, consisting of an estimator taking into account bias and a free-bias estimator, recovered the fault data and was used to localize mobile robot.

6. References

A. Adam, E. Rivlin, and H. Rotstein, "Fusion of fixation and odometry for vehicle navigation," IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, Vol. 29, No. 6, pp. 593-603, 1999.

Y. Bar-Shalom, X. Li, and T. Kirubarajan, Estimation with Applications to Tracking and Navigation, John Wiley & Sons, INC, New York, 2001.

P. Bonnifait, P. Bouron, P. Crubille, and D. Meizel, "Data fusion of four ABS sensors and GPS for an enhanced localization of car-like vehicles," Proceedings of the IEEE International Conference on Robotics and Automation, Seoul, Korea, May 21-26, pp. 1597-1602, 2001.

S. B. H. Bruder, "An information centric approach to heterogeneous multi-sensor integration for robotic applications," Robotics and Autonomous Systems, Vol. 26, No. 4, pp. 255-280, March 1999.

R. Carelli and E. O. Freire, “Corridor navigation and wall-following stable control for sonar-based mobile robots,” Robotics and Autonomous Systems, Vol. 45, No. 3-4, pp. 235-247, December 2003.

N. A. Carlson and M. P. Berarducci, “Federated Kalman filter simulation results,” Journal of the Institute of Navigation, Vol. 41, No. 3, pp. 297-321, 1994.

N. A. Carlson, “Federated square root filter for decentralized parallel processes,” IEEE Transactions on Aerospace and Electronic Systems, Vol. 26, No. 3, pp. 517-525, 1990.

F. Caron, E. Duflos, D. Pomorski, and Ph. Vanheeghe,”GPS/IMU data fusion using multisensor Kalman filtering: introduction of contextual aspects,” Information Fusion, 2005.

M. P. Dana, “Registration: a prerequisite for multiple sensor tracking,” in Multitarget-Multisensor Tracking: Advanced Applications, Y. Bar-Shalom, Ed., Artech House, Norwood, Ma, 1990.

B. Friedland, “Treatment of bias in recursive filtering,” IEEE Transactions on Automatic Control, vol. 14, pp. 359-367, 1969.

J. Guivant, E. Nebot, and S. Baiker, “Localization and map building using laser sensors in outdoor applications,” Journal of Robotic Systems, Vol. 17, No. 10, pp. 565-583, 2000.

J. Huang and H. S. Tan, “A low-order DGPS-based vehicle positioning system under urban environment,” IEEE Transactions on Mechatronics, vol. 11, no. 5, pp. 567-575, 2006.

T. G. Lee, “Centralized Kalman filter with adaptive measurement fusion: its application to a GPS/SDINS integration system with an additional sensor,” International Journal of Control, Automation, and systems, vol. 1, no. 4, pp. 444-452, December 2003.

J. H. Lim and C. U. Kang, “Grid-based Localization of a mobile robot using sonar sensors,” KSME International Journal, Vol. 16, No. 3, pp. 302-309, 2002.

R. Madhavan and H. F. Durrant-Whyte, “Natural landmark-based autonomous vehicle navigation,” Robotics and Autonomous Systems, Vol. 46, pp. 79-95, 2004.

E. M. Nebot and H. Durrant-Whyte, “A high integrity navigation architecture for outdoor autonomous vehicles,” Robotics and Autonomous Systems, Vol. 26, No. 2-3, pp. 81-97, February 1999.

R. Y. Novoselov, S. M. Herman, S. M. Gadaleta, and A. B. Poore, “Mitigating the effects of residual biases with Schmidt-Kalman filtering,” International Conference on Information Fusion, Philadelphia, PA, USA, July 25-29, 2005.

K. Ohno, “A study on outdoor navigation of an autonomous mobile robot based on GPS,” Ph.D. dissertation, Univ. of Tsukuba, Tsukuba, Japan, Mar. 2004.

K. Ohno, T. Tsubouchi, B. Shigematsu, S. Maeyama, and S. Yuta, “Outdoor navigation of a mobile between buildings based on DGPS and odometry data fusion,” Proceedings of the 2003 IEEE International Conference on Robotics and Automation, Taipei, Taiwan, Sep. 14-19, pp. 1978-1984, 2003.

R. Thrapp, C. Westbrook, and D. Subramanian, “Robust localization algorithm for an autonomous campus tour guide,” Proceedings of the 2001 IEEE International Conference on Robotics and Automation, Seoul, Korea, May. 21-26, pp. 2065-2071, 2001.

Y. Zhou, H. Leung, and E. Bosse, “Registration of mobile sensors using the parallelized extended Kalman filter,” Opt. Eng. vol. 36, no. 3, pp. 780-788, March, 1997.

Y. Zhu, Z. You, J. Zhao, K. Zhang, and X. Li, “The optimality for the distributed Kalman filtering fusion,” Automatica, Vol. 37, No. 9, pp. 1489-1493, 2001.

12

Consistent Map Building Based on Sensor Fusion for Indoor Service Robot

Ren C. Luo and Chun C. Lai
Intelligent Robotics and Automation Lab, National Taiwan University
Taipei, Taiwan

1. Introduction

Consider the actual applications of an intelligent service robot (ISR), it is expected that an ISR will not only autonomously estimate the environment structure but also detect the meaningful symbols or signs in the building it services. For example, an ISR has to locate all the docking stations for recharging itself. For an ISR to lead a customer in the department store to any location such as the toy department or the nearest restroom, it must have the essential recognizing and guiding ability for its service. For this purpose, to carry out an applicable self-localization and map building technique for the indoor service robot becomes important and desirable.
In recent years the sensing and computing technology have made tremendous progress. Various simultaneous localization and mapping (SLAM) techniques have been implemented. The principle of SLAM is derived from Bayesian framework. The EKF-SLAM (Durrant-Whyte & Bailey, 2006) is based on robot state estimation. However, EKF-SLAM will fail in large environments caused by inconsistent estimation problem from the linearization process (Rodriguez-Losada et al., 2006) (Bailey et al., 2006) (Shoudong & Gamini, 2007). A full SLAM algorithm is using sequential Monte Carlo sampling method to calculate robot state as particle filter (Montemerlo et al., 2002) (Montemerlo et al., 2003). But the technique will grow exponentially with the increase of dimensions of the state space. Another full scan matching method is suitable for the environment reconstruction (Lu & Milios, 1997) (Borrmann et al., 2008). But the pose variable will also grow enormously depending on the sampling resolution.
Based on the practical needs of a service robot patrol in the building, it is desirable to construct an information map autonomously in a unitary SLAM process. This chapter investigates a consistent map building by laser rangefinder. Firstly, the Covariance Intersection (CI) method is utilized to fuse the robot pose for a robust estimation from wheel encoder and laser scan match. Secondly, a global look up table is built for consistent feature association and a global fitness function is also generated. Finally, the Particle Swarm Optimization (PSO) method is applying to solve the optimal alignment problem. From the proposed method, a consistent map in a unitary localization and mapping process via the sensor fusion and optimal alignment methodology has been constructed and implemented

successfully. Furthermore, a complete 2.5D environment map will be constructed rapidly with the Mesa SwissRanger (Sr, 2006).

2. Robot Pose Estimation

2.1 Covariance Intersection on Sensor Fusion

The Covariance Intersection (CI) is a data fusion algorithm which takes a convex combination of the means and covariance in the information space. The major advantage of CI is that it permits filter and data fusion to be performed on probabilistically defined estimates without knowing the degree of correlation among those estimates. Consider two different pieces of measurement A and B from different sources. If given the mean and variance: $E\{A\} = a$, $E\{B\} = b$, $\mathrm{var}\,\{A, A\} = P_{aa}$, $\mathrm{var}\,\{B, B\} = P_{bb}$, $\mathrm{cov}\,\{A, B\} = P_{ab}$ Define the estimate as a linear combination of A and B where are present the previous estimate of the same target with certain measurement uncertainty. The CI approach is based on a geometric interpretation of the Kalman filter process. The general form of the Kalman filter can be written as:

$$\hat{z} = \omega_a a + \omega_b b \tag{1}$$

$$P_{zz} = \omega_a P_{aa} \omega_a^T + \omega_a P_{ab} \omega_a^T + \omega_b P_{ba} \omega_b^T + \omega_b P_{bb} \omega_b^T \tag{2}$$

where the weights ω_a and ω_b are chosen to minimize P_{zz} .

This form reduces to the conventional Kalman filter if the estimates are independent ($P_{ab} = 0$). The covariance ellipsoid of CI will enclose the intersection region and the estimate is consistent. CI does not need assumptions on the dependency of the two pieces of information when it fuses them. Given the upper bounds $P_{aa} - \overline{P_{aa}} \geq 0$ and $P_{bb} - \overline{P_{bb}} \geq 0$, the covariance intersection estimate output are defined as follows:

$$z = P_{zz}\{\omega_a P_{aa}^{-1} a + \omega_b P_{bb}^{-1} b\} \tag{3}$$

$$P_{zz}^{-1} = \omega_a P_{aa}^{-1} + \omega_b P_{bb}^{-1} \tag{4}$$

where $\omega_a + \omega_b = 1 \quad , \quad 0 \leq \omega_a \ , \omega_b \leq 1$

The parameter ω modifies the relative weights assigned to A and B. Different choices of ω can be used to optimize the covariance estimate with respect to different performance criteria such as minimizing the trace or the determinant of P_{zz} . Let $\alpha \equiv \sqrt{\mathrm{tr}\{\omega_a P_{aa} \omega_a^T\}}$ $\beta \equiv \sqrt{\mathrm{tr}\{\omega_b P_{bb} \omega_b^T\}}$

$$P_{zz} = \frac{\alpha}{\alpha+\beta} P_{aa}^{-1} + \frac{\beta}{\alpha+\beta} P_{bb}^{-1} \tag{5}$$

$$\omega_a = \frac{\alpha}{\alpha+\beta} P_{zz} P_{aa}^{-1} \qquad \omega_b = \frac{\beta}{\alpha+\beta} P_{zz} P_{bb}^{-1} \tag{6}$$

This theorem reveals the nature of the optimality of the best ω in CI algorithm. The CI algorithm also provides a convenient parameterization for the optimal solution in n-square dimensional space. The results can be extended to multiple variables and partial estimates as below:

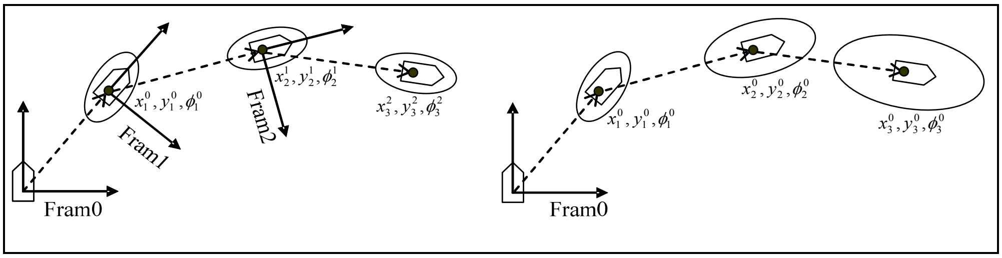
$$z = P_{zz}(\omega_1 A_1^{-1} a_1 + \omega_2 A_2^{-1} a_2 + \ldots + \omega_n A_n^{-1} a_n) \tag{7}$$

$$P_{zz}^{-1} = \omega_1 A_1^{-1} + \omega_2 A_2^{-1} + \omega_3 A_3^{-1} + \ldots + \omega_n A_n^{-1} \tag{8}$$

where $\{a_i, A_i\}$ refers to the i-th measurement input and $\sum_{i=1}^{n} \omega_i = 1$

2.2 Sequence Robot Pose Uncertainty Representation

When a robot platform is moving, the encoder will provide precision pulse resolution for motor control. Unfortunately, the medium between servomotor and floor is not rigid so that errors will occur on robot pose estimation. A Gaussian prior probability may be tried to represent the pose uncertainty from encoder transformation. For a sequence robot pose uncertainty representation, Fig. 1 (a) shows that robot is moving along the dash line.



(a) (b)

Fig. 1. Robot Sequence Pose Estimation and Uncertainty Representation (a) Robot pose uncertainty in opposition to last frame (b) Robot pose uncertainty in opposition to original frame

Each pose uncertainty variation is respect to last local frame or time index. For convenience, the ellipse only represents the Gaussian covariance uncertainty in two-dimension position estimation. The pose (x_3^2, y_3^2, ϕ_3^2) is the mean in the third measurement respect to frame 2 and so on. The problem is that measurement sequence will produce accumulated error respect to original Frame 0 as shown in Fig. 1 (b). A compound mean and error covariance transformation can be derived from previous estimation in expansion matrix form as:

$$\begin{bmatrix} x_3^0 \\ y_3^0 \\ \phi_3^0 \end{bmatrix} = \begin{bmatrix} x_2^1 \cos \phi_1^0 - y_2^1 \sin \phi_1^0 + x_1^0 \\ x_2^1 \sin \theta_1^0 - y_2^1 \cos {}_1^0 + y_1^0 \\ \phi_1^0 + \phi_2^1 \end{bmatrix} \tag{9}$$

$$C_0^3 = J \begin{pmatrix} C_1^0 & 0 \\ 0 & C_2^1 \end{pmatrix} J^T \tag{10}$$

where J is the Jacobian of the transformation at the mean values variables:

$$J = \begin{pmatrix} 1 & 0 & -(y_3^2 - y_1^0) & \cos\phi_1^0 & -\sin\phi_1^0 & 0 \\ 0 & 1 & (x_3^2 - x_1^0) & \sin\phi_1^0 & \cos\phi_1^0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \tag{11}$$

2.3 Pose Estimation from ICP Alignment

In 3D shapes registration application, the iterative closest point (ICP) algorithm was successful apply to align two given point data sets. The ICP algorithm was developed by (Besl & McKay, 1992) and the principle works as follows. Let $P_0 = \{p_1,..., p_m\}$ represent the observation point set and $P_r = \{p_1,..., p_n\}$ be the reference point set. The object of the algorithm is to find a geometric transformation to align the observed point P_0 to the reference point set P_r. This transformation is composed of rotation and translation matrix (R, T). (Nieto et al., 2007) took the algorithm as an association criterion of EKF-SLAM because ICP algorithm makes the association strengthened using the shape as a gate criterion. In this section, the ICP result is regarded as a sensor output on pose estimation between two adjacent measurements from laser ranger. The error covariance evolution on the ICP alignment can be derived as follows:

$$z = \{\rho_i, \theta_i\} \quad , \quad \rho_i = r_i + \varepsilon \tag{12}$$

$$P_i = [\rho_i \cos \theta_i, \rho_i \cos \theta_i]^T \} \ , \ i = 1 \ldots n$$

$$I = \sum_i \left\| (R \cdot p_i^k + T) - map\ (R \cdot p_i^k + T, P^{k-1}) \right\| \tag{13}$$

where k represents the frame or time index and function map( ) maps the data points p_i in frame k into the model points in frame $k - 1$. The ICP algorithm always can find out the transform if the error function can be minimized within a threshold, i.e., ICP arrives in a fit solution. Under this constraint, the covariance approximation depends only on the error function I being minimized and the term $\partial^2 I / \partial Z \partial X$ addresses variation of the error function caused by measurement noise. Therefore, the covariance of pose transformation is represented as:

$$\operatorname{cov}(X) \cong \left(\frac{\partial^2 I}{\partial x^2}\right)^{-1} \frac{\partial^2 I}{\partial Z \partial X} \operatorname{cov}(Z) \frac{\partial^2 I}{\partial Z \partial X} \left(\frac{\partial^2 I}{\partial X^2}\right)^{-1} \tag{14}$$

where Z is from laser measurement and X is the pose transformation. In equation (14), the Cramér–Rao lower bound constraint is proven satisfied (Censi, 2007).

2.4 Multi-Pose Estimation Using CI

For real sensor data acquisition in this study, the sampling rate of encoder is higher than the laser ranger due to higher post-acquisition process requirements of laser ranger. Thus, time sequence alignment is required before fusion process. The encoder uncertainty can be derived by an independent experiment as an approximate covariance matrix. With time sequence index, the uncertainty compound process is needed. When the latest pose estimate is obtained from laser ranger in current time frame, the covariance intersection method will be applied. Fig. 2 shows the concept and the solid ellipse shows the optimal fusion result. In Fig. 3, the actual CI process resulting from robot pose translation is represented. The mobile robot was manipulated to go forward in 50 centimeter (positive y axis on robot local frame) or rotate at each sampling time. The blue line in Fig. 3 shows a pre-determined uncertainty with a 2-deviation on robot x-y translation in each time index. Taking CI fusion with the result from the pre-determined uncertainty of encoder and ICP result from equation (14), the new CI mean is the magenta circle and the magenta line represents the new CI deviation. From the new CI result, a less uncertain interval (the black line) is obtained, i.e., the new estimation will be more close to the true translation (the black star) as shown in Fig. 3.

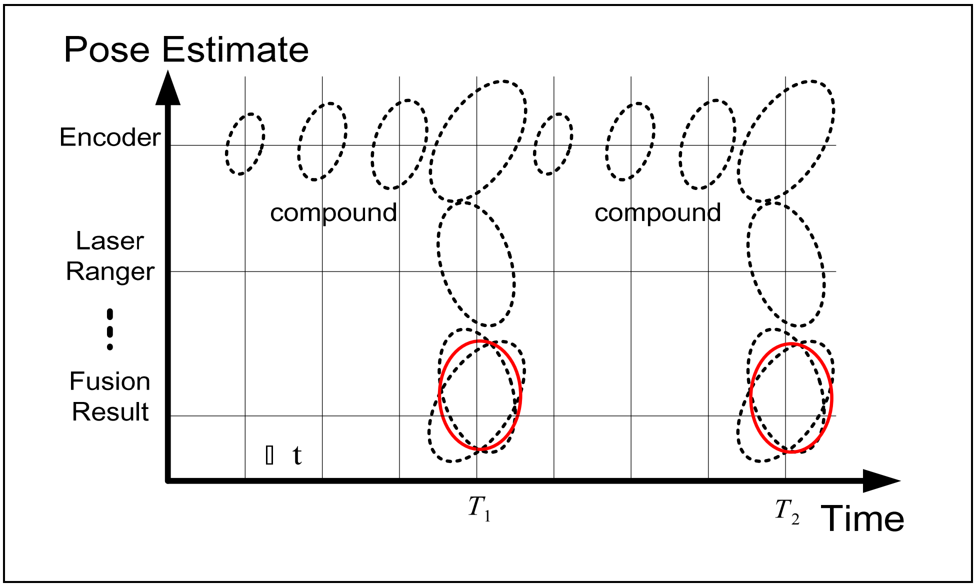


Fig. 2. Position Estimation Using Covariance Intersection

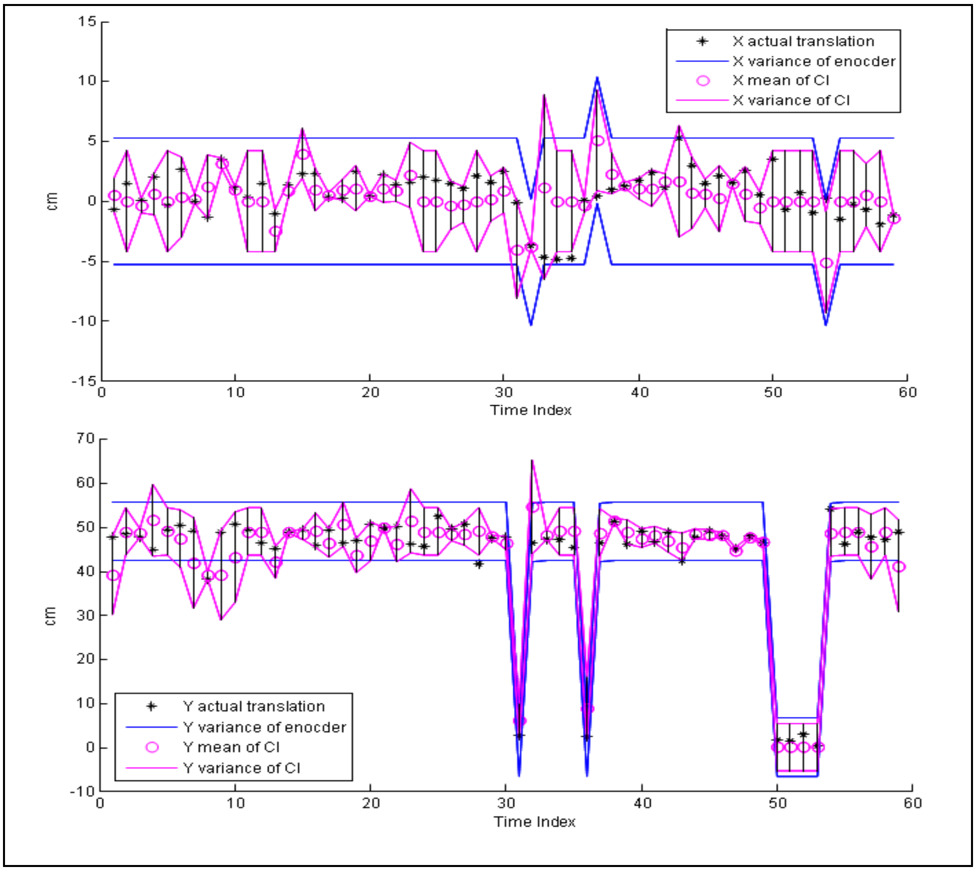


Fig. 3. Covariance Intersection Fusion Result on Robot Pose Translation

3. Consistent Map Alignment

3.1 Segment Extraction from Laser Measurement

For building a consistent geometry map, the distinctive landmarks should be identified and extracted first. Since most of the indoor environment can be efficiently described using polygon segments. As the geometry features are defined based on line segments. From each laser ranger measurement $s = \{p_0, p_1, ..., p_{n-1}, p_n\}$, the Iterative End Point Fit (IEPF) (Borges, & Aldon, 2004) method is applied ahead. The IEPF recursively splits S into two subsets $s_1 = \{p_0, ..., p_j\}$ and $s_2 = \{p_j, ..., p_n\}$ while a validation criterion distance d_{max} is satisfied from point p_j to the segment between p_0 and p_n. Through the iteration, IEPF function will return all segment endpoints $\{p_0, p_j\}$ 、 $\{p_j, p_n\}$. However, IEPF only renders cluster points for each segment as candidate. For more precision line segment estimation, a Linear Regression (LR) method is used to estimate the line equation from each segment candidate.

Fig. 4 (a) shows the laser measurement. In Fig. 4 (b), the starred points are IEPF results and Fig. 4 (c) shows the segment extraction after LR extraction.

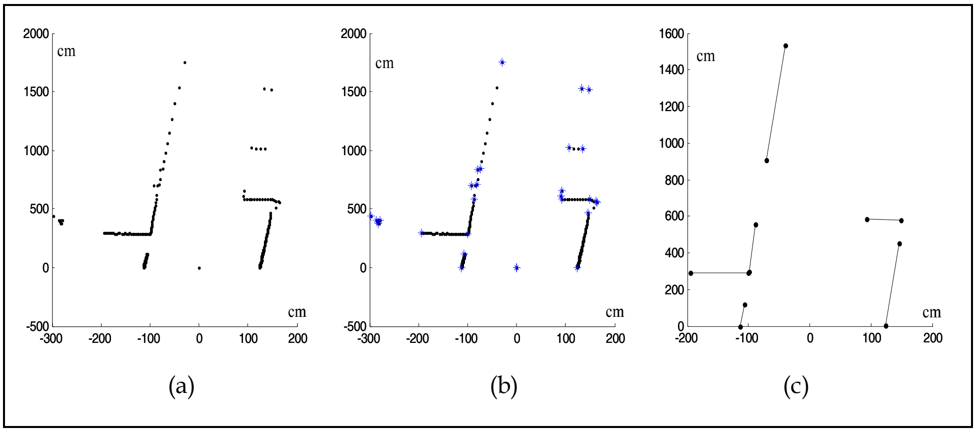


Fig. 4. (a) Laser ranger measurement (b) IEPF result from laser measurement (c) Segment extraction result

3.2 Consistent Association and mapping

The objective of the data association is to eliminate the accumulated error from measurements. The issue is focused on having an accuracy link of landmarks between current and previous observations. From the physical continuity of robot motion, the adjacent measurement of the environment will have the maximum correlation. Also, the ICP method will reach the maximum matching criterion based on the adjacent measurement. Combining encoder measurements in above section, the robust pose estimation is achieved between the adjacent laser measurements. Fig. 5 (a) shows two adjacent laser scans based on robot center. Fig. 5 (b) shows two adjacent laser scans after the pose fusion result.

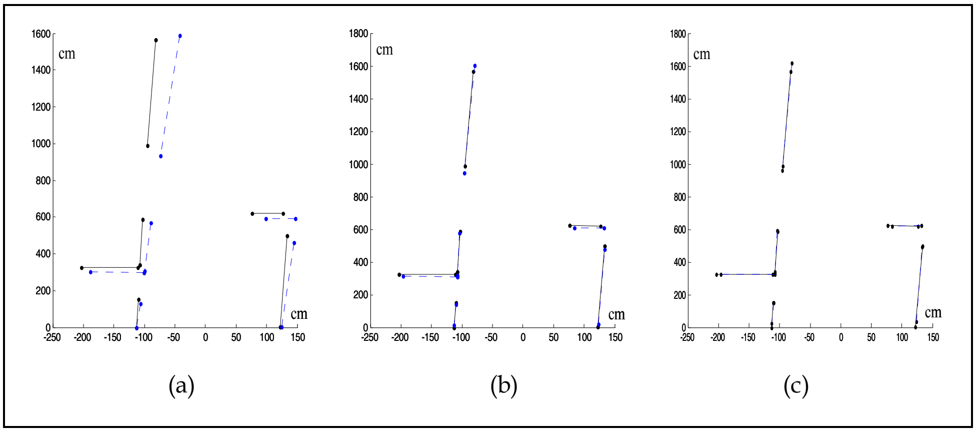


Fig. 5. (a) Segment extraction on two adjacent pose, the solid is model and the dash is new data (b) Fusion result on adjacent pose variation (c) Pose alignment after PSO

If there are *r* solid segments in previous frame *n-1* and there are s dash segments in current frame *n*. A data association criterion is built based on the adjacent segment distance as below:

$$\begin{aligned}
&\text{if } \operatorname{dist}(seg_{i\in r}^{n-1}, seg_{j\in s}^{n}) < \text{threshold} \\
&\quad seg_{j\in s}^{n} \text{ is mapping to } seg_{i\in r}^{n-1} \\
&\text{esle} \\
&\quad seg_{j\in s}^{n} \text{ is a new landmark}
\end{aligned} \tag{15}$$

Via the criterion, the global data association will be connected by successive mapping. Furthermore, the global feature will grow up when a new segment feature is observed. Table I represents the "Association Look Up Table". In measurement frame 0, three segments 1, 2 and 3 are identified and transferred to global features as 1-th, 2-th and 3-th. In frame 1, three segments 1, 2, and 3 are extracted and the first two segments 1 and 2 are mapped to segments 2 and 3 in previous frame 0 by the association criterion. So the segments 1 and 2 in frame 1 will associate to the 2-th and 3-th in global and the segments 3 in frame 1 will create a new landmark as the 4-th in global features. In frame n, there are four segments map to frame n-1 and these four segments are associated in global from the 3-th to the 6-th.

MEASUREMENT FRAME	ADJACENT MAPPING					GLOBAL ASSOCIATION					
0	1	2	3			**1**	**2**	**3**			
1	↗ 1	↗ 2	3				2	3	**4**		
2	↑ 1	↑ 2	↑ 3				2	3	4		
3	↑ 1	↑ 2	↑ 3	4			2	3	4	**5**	
…….	↑ 1	↑ 2	↑ 3	↑ 4	5			3	4	5	**6**
n-1	↗ 1	↗ 2	↗ 3	↗ 4				3	4	5	6
n	↑ 1	↑ 2	↑ 3	↑ 4				3	4	5	6

Table 1. Association Look Up Table

In order to eliminate the residual error accumulated from pose estimation, a global fitness function is generating based on the global association via the association look up table. The fitness function is composed of Euclidean distance between the all segments that associated to the primitive global segments.

$$fitness = \sum_{i=1}^{k} \frac{\left|a_i x_i^1 + b_i y_i^1 + c_i\right| + \left|a_i x_i^2 + b_i y_i^2 + c_i\right|}{\sqrt{a_i + b_i}} \tag{16}$$

where k is the quantity of the segment mapping between the adjacent frames. The a_i, b_i and c_i are the corresponding segment parameters in global frame and (x_i, y_i) are the endpoints which are translated by current robot pose.

3.3 Pose Alignment Using Particle Swam Optimization

The PSO technique was proposed by (Eberhart & Kennedy, 1995) has been widely used in finding solutions for multi-variable optimization problems. Some improvements and applications have also been proposed (Shi & Eberhart, 1998) (Stacey et al., 2003) (Zwe-Lee, 2003). It maintains several particles (each represents a solution) and simulates the behavior of bird flocking to find the final solutions. All the particles continuously move in the search space, toward better solutions, until the termination criteria are met. After certain iterations, the optimal solution or an approximate optimal solution is expected to be found. When applying the PSO technique, each possible solution in the search space is called a particle, which is similar to a bird's move mentioned above. All the particles are evaluated by a fitness function, with the values representing the goodness degrees of the solutions. The solution with the best fitness value for a particle can be regarded as the local optimal solution found so far and is stored as $pBest$ solution for the particle. The best one among all the $pBest$ solutions is regarded as the global optimal solution found so far for the whole set of particles, and is called the $gBest$ solution. In addition, each particle moves with a velocity, which will dynamically change according to $pBest$ and $gBest$. After finding the two best values, a particle updates its velocity by the following equation:

$$\begin{aligned} V_{id}^{new} = \omega \times V_{id}^{old} &+ c_1 \times Rand_1() \times (pBest_{id} - x_{id}) + \\ &c_2 \times Rand_2() \times (gBest_{id} - x_{id}) \end{aligned} \tag{17}$$

where

(a) V_{id}^{new} :the velocity of the i-th particle in the d-th dimension in the next generation;

(b) V_{id}^{old} :the velocity of the i-th particle in the d-th dimension in the current generation;

(c) $pBest_{id}$: the current pBest value of the i-th particle in the d-th dimension;

(d) $gBest_{id}$: the current gBest value of the whole set of particles in the d-th dimension;

(e) x_{id} the current position of the i-th particle in the d-th dimension;

(f) ω : the inertial weight;

(g) c_1 : the acceleration constant for a particle to move to its $pBest$;

(h) c_2 : the acceleration constant for a particle to move to the $gBest_{id}$;

(i) $Rand_1()$, $Rand_2()$: two random numbers between 0 to 1.

After the new velocity is found, the new position for a particle can then be obtained as:

$$x_{id}^{new} = x_{id}^{old} + V_{id}^{new} \tag{18}$$

The proposed approach works well to find out the optimal fitness based on segments alignment. The pose fusion result from encoder and ICP method described in section 2 gives a good initial guess on the optimal search as shown in Fig. 5 (b). Fig. 5 (c) shows the optimal alignment result after PSO. Fig. 6 shows performance comparison of Mathwork Optimal Toolbox (Coleman et al., 1999) and PSO algorithm. Both algorithms were given the same initial guess value with a normalized iteration time. Three fitness functions on two, four and seven alignment conditions are evaluated to find out the global minimum and five particles are predefined in PSO search. The top figure in Fig. 6 shows both optimal algorithms will find out the same global minimum under a 2-alignment fitness constraint. But with more complex fitness condition, the PSO always converge faster and search better than Mathwork function, because the numerical Newton-Raphson technique or direct search (Lagarias et al., 1998) may fall into a local minimum. On the contrary, PSO algorithm has the advantage for global optimal search.

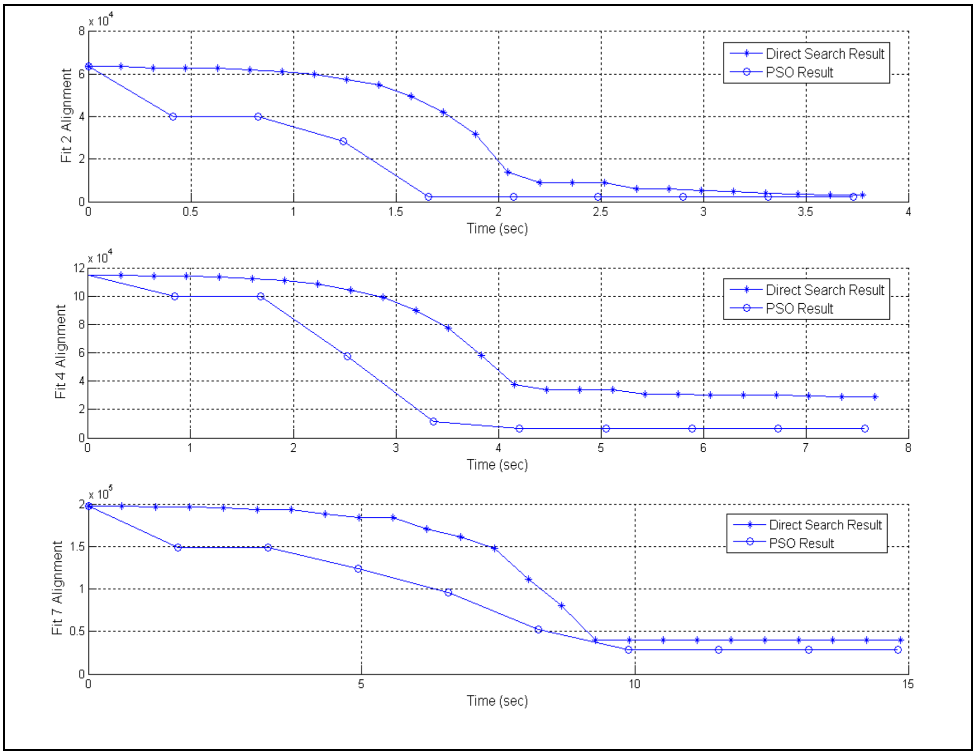


Fig. 6. Performance comparison between Matlab Optimal Toolbox and PSO.

The top shows two segments optimal alignment result, the middle shows four segments optimal alignment result and the bottom shows seven segments optimal alignment result.

4. Consistent Map Building Construction

4.1 Map Building in a Unitary SLAM Tour

A SICK LMS-100 laser ranger is quipped in the robot platform as shown in Fig. 7. In each sampling time, the encoder, laser measurement are compounded and recorded. Applying the consistent alignment methodology with the association look up table described in section 3, the robot pose can be optimally corrected in global frame after each measurement. Fig. 8 shows the complete environment map of the corridor with global segment landmarks and the blue rectangles present the global segment landmarks which are created in the look up table.

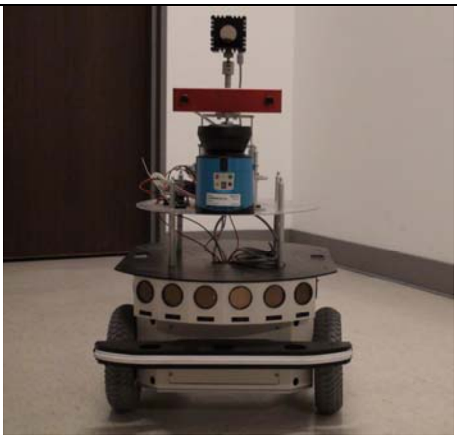


Fig. 7. SICK LMS-100 is equipped on the mobile robot platform

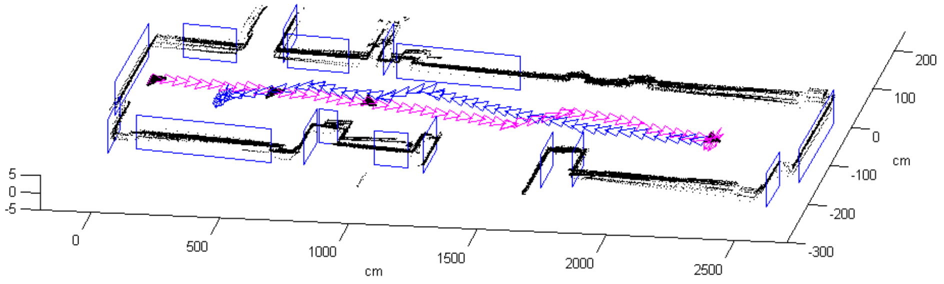


Fig. 8. The consistent map of a corridor with segment landmarks

4.2 Rapid 2.5D Info-Map Building with Mesa SwissRanger

In this experiment, a relief environment map is to be built within a SALM tour and the SwissRagner (Sr, 2006) is applied. SwissRanger belongs to the active 3D sensors. Its measurement is based on a phase-measuring time-of-flight (TOF) principle. A light source

emits a near-infrared wave front that is intensity-modulated with a few tens of MHz. The light is reflected by the scene and imaged by an optical lens onto the dedicated CCD/CMOS sensor with a resolution in 176x144 pixels. Depending on the distance of the target, the captured image is delayed in phase compared to the originally emitted light wave. By measuring the phase delay of each image pixel, it will provide amplitude data, intensity data and distance data, which are weakly correlated to each other. All measurements are being organized by a FPGA hardware implement, which provides an USB interface to access the data values. In practical applications, two parameters will influence the measurement results: one is the integration time and the other is the amplitude threshold. If the integration time is short then the results are very noisy and if it is long then the results are getting blurred with moving objects. A suitable calibration on amplitude threshold is also needed, because the amplitude threshold will filter out the noise due to the reflection of the environment components.
The SwissRanger used in the experiment is equipped with a pre-calibrated SICK laser ranger. Following the previous process, the robot pose will be estimated with the consistent map alignment in each time index. With the pre-calibrated SICK laser ranger, the 3D measurements from SwissRanger are available. To execute a calibrated transformation, all the 3D measurements will also keep on the optimal alignment. Fig. 9 shows the complete 2.5D info-map building result with the SwissRanger scanning.

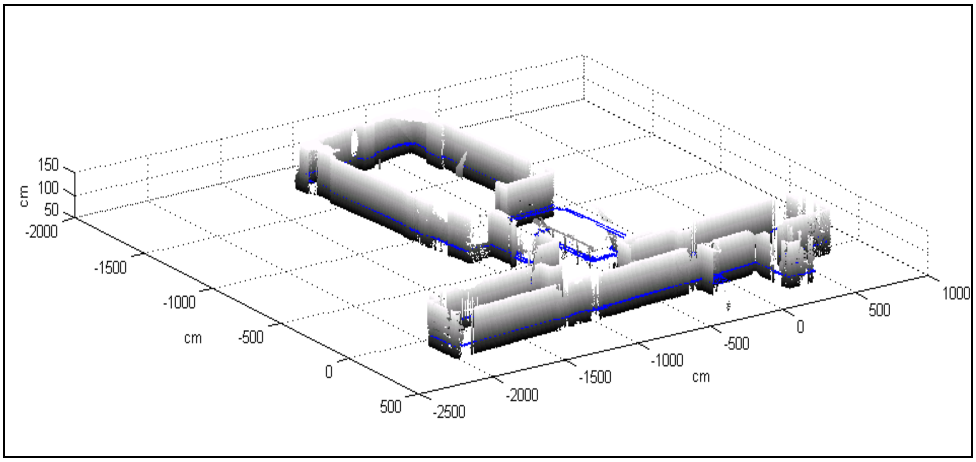


Fig. 9. A complete 2.5D map building with SwissRanger scanning

5. Conclusion

This chapter presents a consistent map construction in a unitary SLAM (simultaneously localization and mapping) process through the sensor fusion approach and optimal alignment technologies. The system will autonomously provide the environment geometrical structure for intelligent robot service in a building. In order to build the consistent map, a CI (Covariance Intersection) rule fuses the uncertainty from wheel encoder and ICP (Iterative Closest Point) result as a robust initial value of the fitness function. The alignment fitness function is composed of the Euclidean distances which are associated from current features to the global ones by a quick look up table. After applying the artificial PSO (Particle Swarm Optimization) algorithm, the optimal robot pose for the map alignment is obtained. Finally, by employing SwissRanger sensor, a complete 2.5D info-map cab be rapidly built up for indoor service robot applications.

6. References

Bailey, T., Nieto, J., Guivant, J., Stevens, M., & Nebot, E. (2006). Consistency of the EKF-SLAM Algorithm, *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on,* pp. 3562-3568.

Besl, P. J., & McKay, H. D. (1992). A method for registration of 3-D shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* 14, 239-256.

Borges, G. A., & Aldon, M. J. (2004). Line extraction in 2D range images for mobile robotics. *Journal of Intelligent and Robotic Systems,* 40, 267-297.

Borrmann, D., Elseberg, J., Lingemann, K., Nuchter, A., & Hertzberg, J. (2008). Globally consistent 3D mapping with scan matching. *Robotics and Autonomous Systems,* 56, 130-142.

Censi, A. (2007). On achievable accuracy for range-finder localization, *Robotics and Automation, 2007 IEEE International Conference on,* pp. 4170-4175.

Coleman, T., Branch, M. A., & Grace, A. (1999). Optimization toolbox. *For Use with MATLAB. User's Guide for MATLAB 5, Version 2, Relaese II.*

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis,* New York.

Durrant-Whyte, H., & Bailey, T. (2006). Simultaneous localization and mapping: part I. *Robotics & Automation Magazine, IEEE,* 13, 99-110.

Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory, *Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on,* pp. 39-43.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization, *Neural Networks, 1995. Proceedings., IEEE International Conference on,* pp. 1942-1948.

Lagarias, J. C., Reeds, J. A., Wright, M. H., & Wright, P. E. (1998). Convergence properties of the Nelder-Mead simplex algorithm in low dimensions. *SIAM Journal on Optimization,* 9, 112-147.

Lu, F., & Milios, E. (1997). Robot pose estimation in unknown environments by matching 2d range scans. *Journal of Intelligent and Robotic Systems,* 18, 249-275.

Montemerlo, M., Thrun, S., Koller, D., & Wegbreit, B. (2002). FastSLAM: A factored solution to the simultaneous localization and mapping problem, pp. 593-598, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Montemerlo, M., Thrun, S., Koller, D., & Wegbreit, B. (2003). FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges, pp. 1151-1156), LAWRENCE ERLBAUM ASSOCIATES LTD.

Nieto, J., Bailey, T., & Nebot, E. (2007). Recursive scan-matching SLAM. *Robotics and Autonomous Systems,* 55, 39-49.

Rodriguez-Losada, D., Matia, F., & Galan, R. (2006). Building geometric feature based maps for indoor service robots. *Robotics and Autonomous Systems,* 54, 546-558.

Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer, *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on,* pp. 69-73.

Shoudong, H., & Gamini, D. (2007). Convergence and Consistency Analysis for Extended Kalman Filter Based SLAM. *Robotics, IEEE Transactions on,* 23, 1036-1049.

Sr, C. S. 3000 manual v1. 02, June 2006. *Available to Swiss-Ranger customers: http://www. swissranger. ch/customer*.

Stacey, A., Jancic, M., & Grundy, I. (2003). Particle swarm optimization with mutation, *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on,* pp. 1425-1430.

Zwe-Lee, G. (2003). Discrete particle swarm optimization algorithm for unit commitment, *Power Engineering Society General Meeting, 2003, IEEE,* pp. 424.

Mobile Robot Localization and Map Building for a Nonholonomic Mobile Robot

Songmin Jia[1] and AkiraYasuda[2]
[1]*Beijing University of Technology*
Beijing, 100124,P.R.CHINA
[2]*Panasonic Electric Works Co., Ltd. 2009*

1. Introduction

Mobile robot localization and map building are very important for a mobile robot to perform a navigation-based service task to aid the aged or disabled in an indoor environment (office, facilities or at home). In order to enable mobile robot to perform service tasks autonomously, the information of environment in which the mobile robot moves is needed. To obtain the accurate map, many different sensors system and techniques have been developed. Approaches of mapping can roughly be classified according to the kind of sensor data processed and the matching algorithms [S. Thrun et al., 1996, 1998, 2000, 2004; Chong, K.S et al., 1996; C.-C. Wang et al., 2002; G. Dissanayake et al., 2000; B. barshan et al., 1995; Weimin Shen et al., 2005]. In this paper, we proposed a method of map building using interactive GUI for a mobile robot. The reason we proposed this method is that it is difficult for a mobile robot to generate an accurate map although many kinds of sensors are used. The proposed method enables the operator to modify map built by sensors, compared with the real-time video from web camera by using modification tool in interactive GUI. Laser Range Finder (LRF) was used to get the information of the environment in which mobile robot moving. In order to improve self-localization of mobile robot, Extended Kalman Filter (EKF) was introduced. By the results of simulation and experiments, the developed EKF is effective in self-localization of mobile robot. This paper introduces the architecture of the system and gives some experimental results to verify the effectiveness of the developed system.
The rest of the paper consists of 5 sections. Section 2 describes the structure of the proposed system. Section 3 presents the developed algorithm of self-localization of mobile robot and gives some simulation and experimental results to verify the effectiveness of the EKF in improving precision of positioning of mobile robot. Section 4 details the developed LRF data processing algorithm. Section 5 introduces the developed interactive GUI. The experimental results are given in Section 6. Section 7 concludes the paper.

2. System Description

Figure 1 illustrates the architecture of the proposed system. It includes a nonholonomic mobile robot, Laser Ranger Finder (LRF), web camera, robot controller, data processing PC and GUI for operator. Data from LRF, odometry and real-time video from web camera are processed on data processing PC, and map is built according to the processed results. These results are transferred to the GUI PC and shown in GUI via Internet.

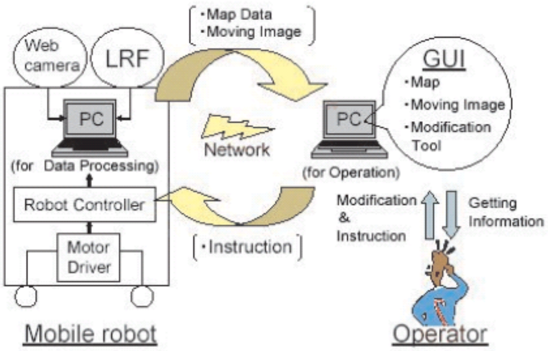


Fig. 1. The developed system architecture.

In the previously developed system, an omnidirectional mobile robot was used to perform service tasks. Owing to the specific structure of its wheel arrangement, it is difficult for a mobile robot to pass over a bump or enter a room where there is a threshold. Another important point is to lower costs and decrease the number of motors so that the battery can supply enough electricity for a mobile robot to run for a longer time. Figure 2 illustrates the developed mobile robot platform. In our new system, we developed a non-holonomic mobile robot that was remodelled from a commercially available manual cart. The structure of the front wheels was changed with a lever balance structure to make the mobile robot move smoothly and the motors were fixed to the two front wheels. It has low cost and can easily pass over a bump or gap between the floor and rooms. We selected the Maxon EC motor and a digital server amplifier 4-Q-EC 50/5 which can be controlled via RS-232C. For the controller of the mobile robot, a PC104 CPU module (PCM-3350 Geode GX1-300 based) is used, on which RT-Linux is running. For communication between the mobile robot and the mobile robot control server running on the host computer, a wireless LAN (PCMCIA-WLI-L111) is used.

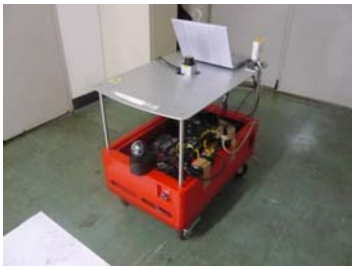


Fig. 2. The developed mobile robot platform.

Laser Ranger Finder is a precision instrument based on Time of Flight principle. URG-X04LX (HOKUYO AUTOMATIC CO., LTD) was used in our system to detect the environment. It uses IR laser (wave length 785mm). Its distances range is about 60 to 4095mm 0-240°. The measurement error is about ± 10mm between 60 to 1000mm and 10% between 1000-4095mm. The scanning time for one circle is about 100ms.
The QuickCam Orbit cameras were used in our system to transfer the real-video for the operator with automatic face tracking and mechanical Pan, Tilt and face tracking feature. It mechanically and automatically turns left and right for almost a 180-degree horizontal view or up and down for almost 90 degree top-to-bottom view. It has high-quality videos at true CCD 640×480 resolution and its maximum video frame rate are 30 fps (frames per second) and work with both USB 2.0 and 1.1.

3. Self-Localization of Mobile Robot by Extended Kalman Filter

Odometry is usually used in relative position of mobile robot because it is simple, inexpensive and easily accomplished in real time, but it fails to accurately positioning over long traveling distance because of wheel slippages, mechanical tolerances and surface roughness. We use Extended Kalman Filter to improve self-localization for mobile robot. The Extended Kalman Filter algorithm [Greg Welch et al., 2004] is recursive and only the current state calculation will be used, so it is easy to implement. In order to use EKF, we have to build system state model and sensor model. The movement model of mobile robot used in our system was shown in Figure 3. v_{Rk}, v_{Lk} are the speed of right wheel and left wheel, l is the distance between two wheels, w_k is the system error, and assumed to be white Gaussian with covariance W_k. The movement model of mobile robot can be defined as equations (1)-(6), and the state model can be defined as equation (7).
Equations (8)-(12) are the observation model. LRF detects wall or corner of environment as Landmarks for observation model. Equations (15)-(21) give the recurrence computation for predetecting and updating of the state of the mobile robot.

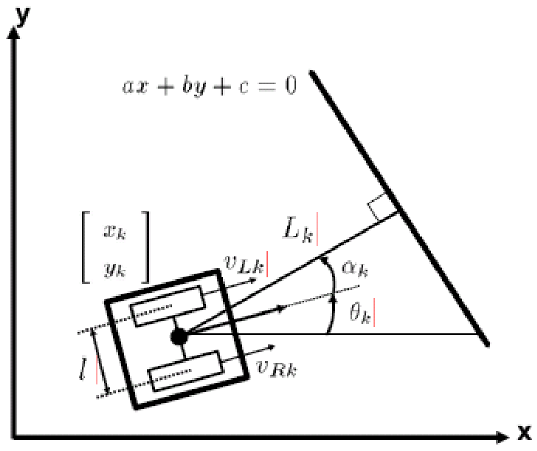


Fig. 3. Movement model of mobile robot platform.

$$\mathbf{x_k} = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} \tag{1}$$

$$\mathbf{u_k} = \begin{bmatrix} v_{Rk} \\ v_{Lk} \end{bmatrix} \tag{2}$$

$$x_{k+1} = x_k + \frac{v_{Rk} + v_{Lk}}{2} cos\theta_k \cdot \Delta t \tag{3}$$

$$y_{k+1} = y_k + \frac{v_{Rk} + v_{Lk}}{2} sin\theta_k \cdot \Delta t \tag{4}$$

$$\theta_{k+1} = \theta_k + \frac{v_{Rk} - v_{Lk}}{l} \cdot \Delta t \tag{5}$$

$$f(\mathbf{x_k}, \mathbf{u_k}) = \begin{bmatrix} x_k + \frac{v_{Rk}+v_{Lk}}{2} cos\theta_k \cdot \Delta t \\ y_k + \frac{v_{Rk}+v_{Lk}}{2} sin\theta_k \cdot \Delta t \\ \theta_k + \frac{v_{Rk}-v_{Lk}}{l} \cdot \Delta t \end{bmatrix} \tag{6}$$

$$\mathbf{x_{k+1}} = \mathbf{f}(\mathbf{x_k}, \mathbf{u_k}) + \mathbf{w_k} \tag{7}$$

$$\mathbf{y_k} = \begin{bmatrix} L_k \\ \alpha_k \end{bmatrix} \tag{8}$$

$$L_k = \frac{\mid ax_k + by_k + c \mid}{\sqrt{a^2 + b^2}} \tag{9}$$

$$\alpha_k = tan^{-1}(-\frac{a}{b}) - \frac{\pi}{2} - \theta_k \tag{10}$$

$$h(\mathbf{x_k}) = \begin{bmatrix} \frac{|ax_k+by_k+c|}{\sqrt{a^2+b^2}} \\ tan^{-1}(-\frac{a}{b}) - \frac{\pi}{2} - \theta_k \end{bmatrix} \tag{11}$$

$$\mathbf{y_k} = \mathbf{h}(\mathbf{x_k}) + \mathbf{v_k} \tag{12}$$

$$\mathbf{v_k} \sim \mathbf{N}(\mathbf{0}, \mathbf{V_k}) \tag{13}$$

$$\mathbf{V_k} = \begin{bmatrix} {\sigma_L}^2 & 0 \\ 0 & {\sigma_\alpha}^2 \end{bmatrix} \tag{14}$$

System state one-step prediction $\hat{x}_{k+1}$ and its associated covariance $\hat{P}_{k+1}$ can be calculated from previous k and previous $\hat{x}_k$ and $\hat{P}_k$ (equation (15)-(16)). The state update can be done using equation (19)-(21).

$$\hat{\mathbf{x}}_{\mathbf{k+1}}(-) = \mathbf{f}(\hat{\mathbf{x}}_{\mathbf{k}}(+), \mathbf{u_k}) \tag{15}$$

$$\hat{\mathbf{P}}_{\mathbf{k+1}}(-) = \mathbf{F_k}\hat{\mathbf{P}}_{\mathbf{k}}(+)\mathbf{F_k^T} + \mathbf{G_k}\mathbf{W_k}\mathbf{G_k^T} \tag{16}$$

Here, $\hat{x}_k(+)$ is state estimation at t = k after observation. $\hat{x}_{k+1}$ is state estimation at t = k +1 before observation. $\hat{P}_k(+)$ is error covariance at t = k after observation. $\hat{P}_{k+1}(-)$ is error covariance at t = k+1 before observation. F_k, G_k can be calculated by the following equation.

$$\mathbf{F_k} = \frac{\partial \mathbf{f}(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}} |_{\mathbf{x}=\hat{\mathbf{x}}_\mathbf{k}(+),\mathbf{u}=\mathbf{u_k}} \tag{17}$$

$$\mathbf{G_k} = \frac{\partial \mathbf{f}(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}} |_{\mathbf{x}=\hat{\mathbf{x}}_\mathbf{k}(+),\mathbf{u}=\mathbf{u_k}} \tag{18}$$

$$\mathbf{K_{k+1}} = \hat{\mathbf{P}}_\mathbf{k+1}(-)\mathbf{H}^\mathbf{T}_\mathbf{k+1}(\mathbf{H_{k+1}}\hat{\mathbf{P}}_\mathbf{k+1}(-)\mathbf{H}^\mathbf{T}_\mathbf{k+1} + \mathbf{V_{k+1}}) \tag{19}$$

$$\hat{\mathbf{x}}_\mathbf{k+1}(+) = \hat{\mathbf{x}}_\mathbf{k+1}(-) + \mathbf{K_{k+1}}(\mathbf{y_{k+1}} - \mathbf{h}(\hat{\mathbf{x}}_\mathbf{k+1}(-))) \tag{20}$$

$$\hat{\mathbf{P}}_\mathbf{k+1}(+) = (\mathbf{I} - \mathbf{K_{k+1}}\mathbf{H_{k+1}})\hat{\mathbf{P}}_\mathbf{k+1}(-) \tag{21}$$

Figure 4 illustrates the flowchart of the developed method of map building. First the robot system gets the information of environment of mobile robot moving in by LRF (distance and angle), then transforms the coordinates of the processed results of LRF data, and draws the results in local map. Using these results, the robot system can predict the $\hat{x}_{k+1}(-)$ using Extended Kalman Filter, get observation parameters and update $\hat{x}_{k+1}(+)$. Lastly, the robot system will transform coordinates of the results from local to global. If there are some errors in the map built by sensors, the operator can modify the map generated using modification tool in interactive GUI by comparing the processing results of sensors and real-time video.

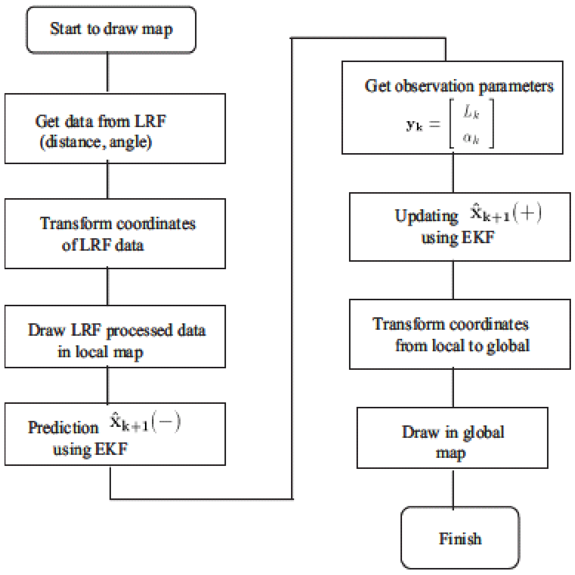

Start to draw map
Get data from LRF (distance, angle)
Transform coordinates of LRF data
Draw LRF processed data in local map
Prediction $\hat{x}_{k+1}(-)$ using EKF
Get observation parameters $y_k = \begin{bmatrix} L_k \\ \alpha_k \end{bmatrix}$
Updating $\hat{x}_{k+1}(+)$ using EKF
Transform coordinates from local to global
Draw in global map
Finish

Fig. 4. Flowchart of the developed method of map building.

We have done simulations and experiments to compare the results of using Extended Kalman Filter and odometry. Figure 5 illustrates the some results of simulations and Figure 6 illustrates the experimental results of using developed mobile robot. The green line is the real trajectory of the mobile robot, the red line is the detected trajectory by using Extended Kalman Filter and the blue line is the trajectory detected by just odometry. According to these results, we know that the developed Extended Kalman Filter can improve the precision of self-localization of mobile robot.

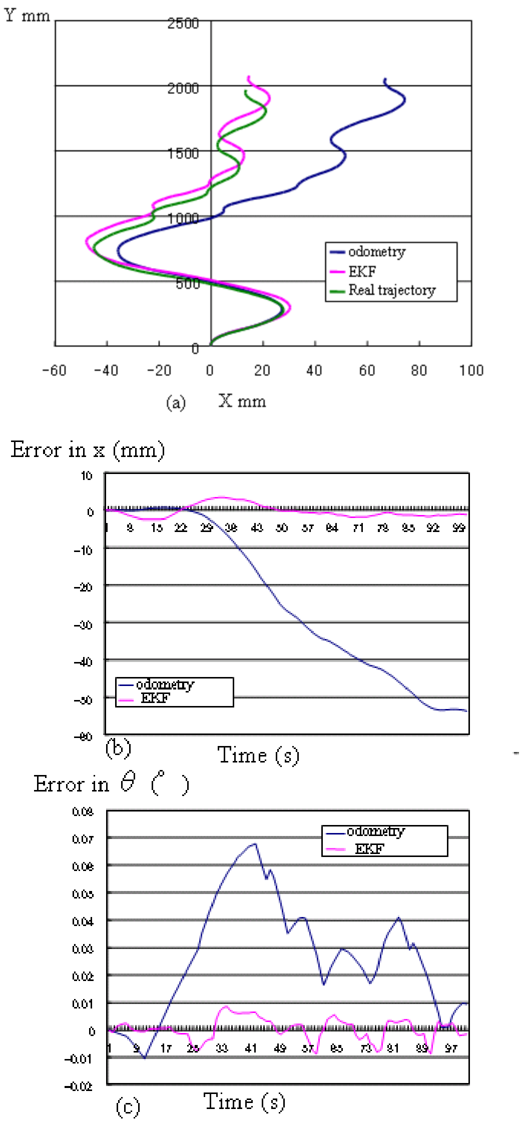


Fig. 5. Simulation results of using EKF.

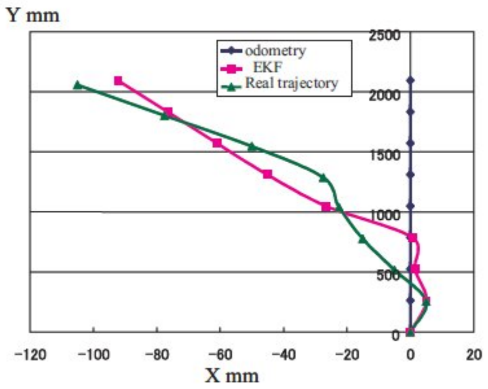
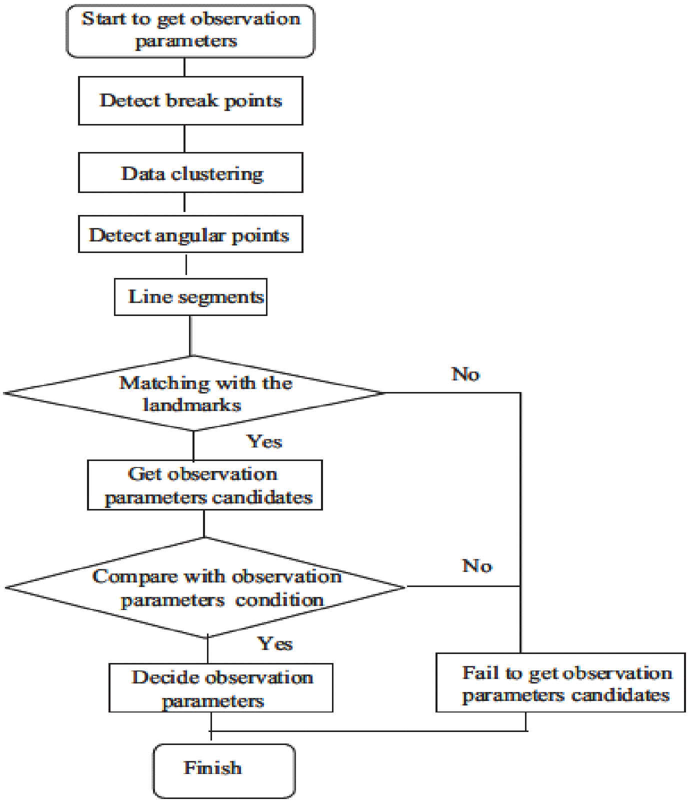


Fig. 6. Experimental results of using EKF.

4. LRF Data Processing Algorithm



Start to get observation parameters
Detect break points
Data clustering
Detect angular points
Line segments
Matching with the landmarks
No
Yes
Get observation parameters candidates
Compare with observation parameters condition
No
Yes
Decide observation parameters
Fail to get observation parameters candidates
Finish

Fig. 7. Flowchart of LRF processing data.

It is necessary to process scanning data from LRF in order to get observation model parameters for map building of mobile robot. We proposed processing algorithm. Figure 7 illustrates the flowchart of the processing LRF data algorithm. It includes:

- Detect break points
- Data clustering
- Angular point's detection
- Line segment
- Match with landmarks
- Decide observation parameters candidate
- Decide observation parameters

4.1 Detect Break Points

Break points can be detected [Zhuang Yan et al., 2004] using the following equation (22) for total of 768 points of laser reading set per scan, here T_{BP} is the threshold given by experimental results in advance. Figure 8 illustrates the principle of detecting the break points.

$$|P_{i+1} - P_i| > T_{BP}, i = 0, ..., 766 \tag{22}$$

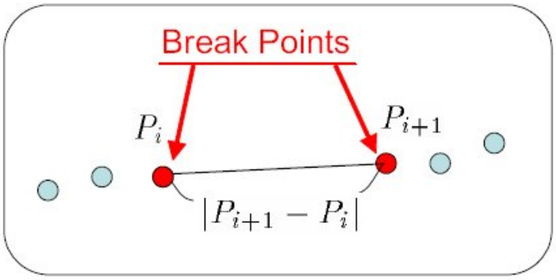


Fig. 8. Break Points detection.

4.2 Data Clustering

After break points were detected, we classified data into different clusters. If the data in one cluster (defined as N_{BP}) is smaller than T_{NBP} (threshold given by experimental results), the line segment in next section will be not done because the accurate results cannot be got.

4.3 Angular Points Detection

Figure 9 illustrates how to detect angular points. For the points P_i, ..., P_{i+n} in each cluster, the slop from P_i to $P_{i+(n/2)}$ and from $P_{i+(n/2)}$ to P_{i+n} are calculated using the least squire method. If the ϕ satisfies the following condition, $P_{i+(n/2)}$ is detected as angular points. Here, T_{AP} is threshold given by experimental results.

$$|\phi - \frac{\pi}{2}| < T_{AP} \tag{23}$$

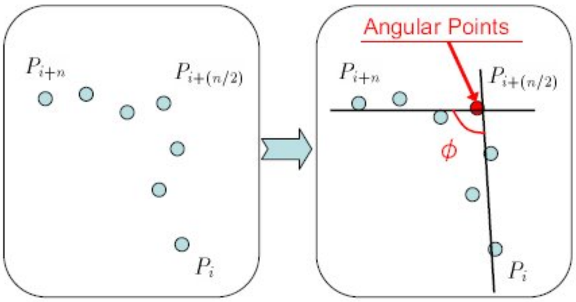


Fig. 9. Angular Points detection

4.4 Line Segment

Using the angular points for each cluster, line features are extracted. Assume that the line expression is given by $y = g_G x + i_G$, parameters g_G and i_G are calculated by the least square method. Figure 10 illustrates the flow of the line segment.

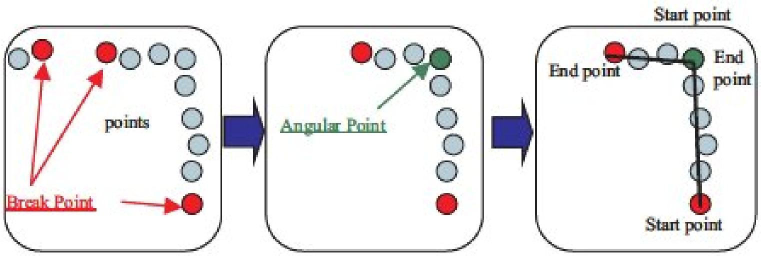


Fig. 10. Line segment.

4.5 March with Landmarks

Line extracted by above section was used to match with the landmarks. If the lines extracted satisfy the following conditions, matching is successful, otherwise matching failed. Here, T_g, T_i, $T_{b=0}$ are threshold.

$$|-\frac{a}{b} - g_G| < T_g \tag{24}$$

$$|-\frac{c}{b} - i_G| < T_i \tag{25}$$

In the case of $b = 0$, the condition will be:

$$|-\frac{c}{a} - i_G| < T_{b=0} \tag{26}$$

4.6 Decide Observation Parameters Candidate

For the lines that matching were successful, we calculated $\mathbf{y}_{k(candidate)} = [L_{k(candidate)}\ \alpha_{k(candidate)}]^{T}$ as observation parameters candidate.

4.7 Decide Observation Parameters

If there are multi-lines for one landmark, we have to extract one accurate line as observation model parameters from them. First the observation model parameters are calculated by each line extracted and if $L_{k(h)}$, $\alpha_{k(h)}$ satisfy the following condition, the observation parameters can be decided.

$$|\alpha_{k(min)} - \alpha_{k(h)})| < T_{\alpha} \tag{27}$$

$$|L_{k(min)} - L_{k(h)})| < T_{L} \tag{28}$$

Figure 11 illustrates some samples of processing LRF data. Figure 11(a) is data description from LRF; Figure 11(b) is the sample of detecting break points; Figure 11 (c) is the sample of detecting angular points; Figure 11 (d) is the sample of matching with landmarks.

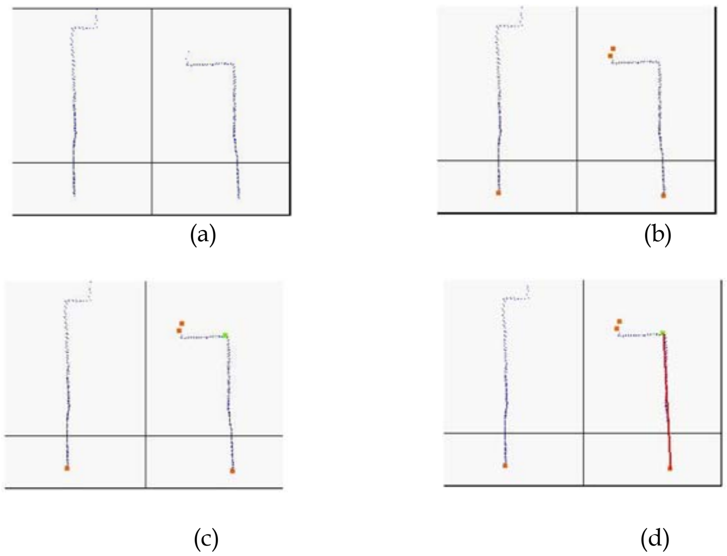


Fig. 11. LRF data processing. (a) is data description; (b) is the sample of detecting break points; (c) is the sample of detecting angular points; (d) is the sample of matching with landmarks.

5. The Developed Interactive GUI

As described before, the reason we proposed this map building method is that it is difficult for a mobile robot to generate an accurate map although many kinds of sensors are used. We proposed map building method that enables the operator to modify map built by sensors, compared with the real-time video from web camera by using interactive GUI,

which can improve the precision of map building and simplify the system. Figure 12 illustrates the developed interactive GUI for the system. It includes:
• Display the map generated by sensors.
• Display the real environment of the mobile robot moving in.
• Pan and tilt control part for camera.
• Direct control function for mobile robot.
• Modification tool for the operator to modify the map.

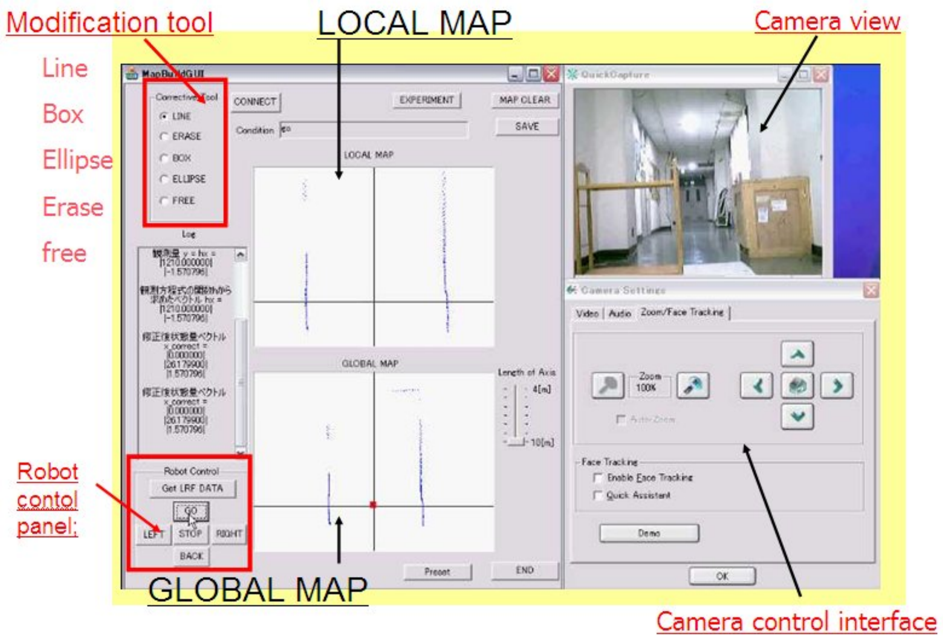


Fig. 12. The developed interactive GUI.

5.1 Map Display

Map first was generated from distance and angle data from LRF and odmetry from robot controller. The upper part is local area map about 4m×4m to display current map around the mobile robot in order for operator to observe accurately. The lower part is global map about 10m×10m to display all map built from the start moving position.

5.2 Display the Real Environment by Web Camera

In order for operator to monitor the real environment of the mobile robot moving and to modify the error of map, the QuickCam Orbit web camera was used to transfer real-time video and displayed in GUI. There are also pan, tilt and zoom modification control of camera in GUI to enable the operator to confirm the environment around the mobile robot.

5.3 Modification Tool for User to Modify the Map

Modification tool enables the operator to modify the error of map built by LRF and odemetry. Modification tool of map includes ″Line, Box, Ellipse, Free, Erase″ that enables the operator to modify the map generated by sensors if there are some errors in map built by sensors. For example, the operator can use ″Box″ in modification tool to draw the box in the map to modify the error of misrecognition of the bookshelf in the map because LRF just detected the feet part of bookshelf. Additionally, there are also the direct control function such as ″move back, move forward, turn right, turn left, stop″ to mobile robot in developed interactive GUI in order for operator to control mobile robot.

6. Experimental Results

We have done some simulations and experiments using EKF to improve the localization of mobile robot and the results verified that using EKF can enhance the precision of the localization of mobile robot. We have also done some experiments of map building using the proposed method, and some experimental results were shown in Figure 13 and Figure 14. Figure 13(a)-(d) are samples of the mobile robot moving for mapping; Figure 14(b) is the designed GUI. When the mobile robot platform moves, the map will simultaneously built by sensors and shown in GUI. The operator can modify the map generated by modification tool if there were some errors by comparing with real-time video from web camera. Figure 14(c) illustrated there was error in map because LRF cannot detect the black part on the wall and Figure 14(d) illustrated modified results. Figure 14(e) illustrates some errors because LRF just detected the feet part of bookshelf and Figure 14(f) was modified results by the operator using modification tool in GUI. The experimental results verified the effectiveness of the proposed method.

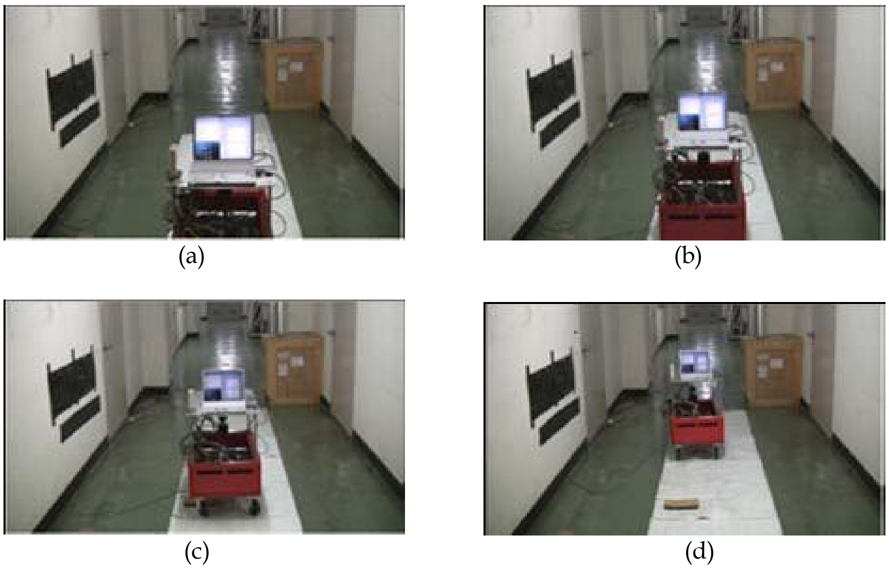


Fig. 13. Some images of experiments of mobile robot moving to build map.

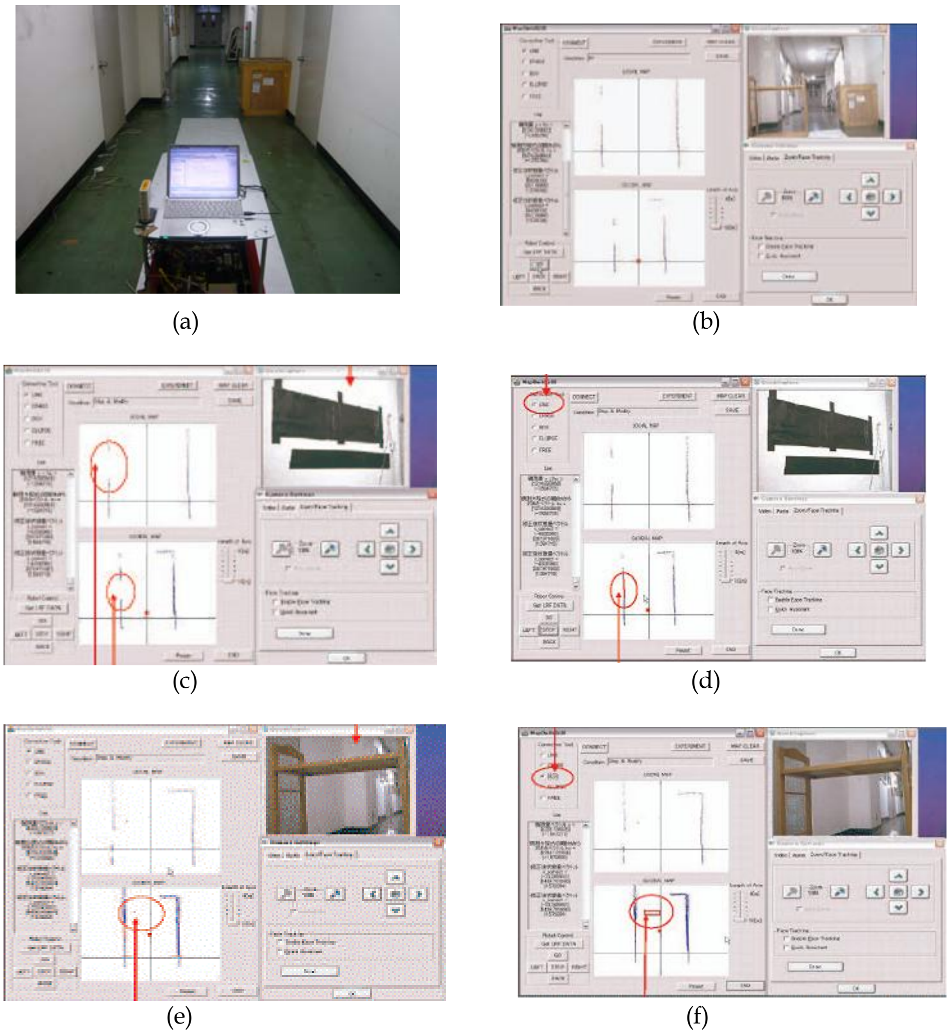


Fig. 14. Some images of experiments using interactive GUI to modify the map built by LRF.

7. Conclusion

Map building is a very important issue for mobile robot performing a task. It was difficult to generate an accurate map although many sensors or complicated system were used. This paper proposed a method of map building using interactive GUI for an indoor service mobile robot. The developed system enabled the operator to modify map built by LRF, compared with the real-time video from web camera if there was some errors in map because of misrecognition of sensors. We developed Extended Kalman Filter to improve the precision of the self-localization of mobile robot. The simulation and experimental results verified the effectiveness of the developed system. The current system just enabled the local user to control; developing network function to allow the remote user to access system is the topic in the future.

8. References

B. barshan and H.F. Durrant-Whyte, Inertial sensing for mobile robotics, IEEE Trans. Robotics and Automation 3 (1995), 328.

J. Borenstein and L.Fenf, Measurement and correction of systematic odeometry errors in mobile robots, IEEE J. Robotics and Automation 6 (1996), 869.

C.-C. Wang and C. Thorpe. Simultaneous localization and mapping with detection and tracking of moving objects. In Proc. of the IEEE International Conference on Robotics and Automation (ICRA), 2002.

Chong, K.S. and Kleeman, L. 1996. Mobile robot map building from an advanced sonar array and accurate odometry. Technical Report MECSE- 1996-10, Melbourne. Monash University. Department of Electrical and Computer Systems Engineering.

G. Dissanayake, H. Durrant-Whyte, and T. Bailey. A computationally efficient solution to the simultaneous localization and map building (SLAM) problem. In ICRA’2000 Workshop on Mobile Robot Navigation and Mapping, 2000.

Seung-Hun Kim, Chi-Won Roh, Sung-Chul Kang and Min-Yong Park,“Outdoor Navigation of a Mobile Robot Using Differential GPS and Curb Detection”, In Proceedings of 2007 IEEE International Conference on Robotics and Automation, pp. 3414-3418.

Zhuang Yan, Tang Shubo, Liu Lei and Wang Wei,“Mobile Robot Indoor Map Building and Pose Tracking Using Laser Scanning”, In Proceeding of the 2004 International Conference on Intelligent Mechatronics and Automation Chengdu, China August 2004.

S. Thrun, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Minerva: A secondgeneration museum tour-guide robot. In Proceedings of IEEE International Conference on Robotics and Automation (ICRA'99), Detroit, Michigan, May 1999.

S. Thrun, W. Burgard, and D. Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. Machine Learning and Autonomous Robots (joint issue), 31(5):1{25, 1998.

S. Thrun, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping. In IEEE International Conference on Robotics and Automation (ICRA 2000), San Francisco, CA, April 2000.

S. Thrun, A. Bucken, Integrating grid-based and topological maps for mobile robot navigation, in: Proc. of the 13th National Conference on AI, pp.128-133(1996). Greg Welch and Gary Bishop,“An Introduction to the Kalman Filter”, UNC-Chapel Hill, TR 95-041, April 5,2004.

Weimin Shen and Jason Gu, “Fuzzy Approach For Mobile Robot Positioning”, In Proceedings of 2005 IEEE International Conference on Intelligent Robots and Systems, pp2906-2911

14

Robust Global Urban Localization Based on Road Maps

Jose Guivant, Mark Whitty and Alicia Robledo
School of Mechanical and Manufacturing Engineering
The University of New South Wales
Australia

1. Introduction

This paper presents a method to perform global localization in urban environments using segment-based maps in combination with particle filters. In the proposed approach the likelihood function is generated as a grid, derived from segment-based maps. The scheme can efficiently assign weights to the particles in real time, with minimum memory requirements and without any additional pre-filtering procedure. Multi-hypothesis cases are handled transparently by the filter. A local history-based observation model is formulated as an extension to deal with 'out-of-map navigation cases. This feature is highly desirable since the map can be incomplete, or the vehicle can be actually located outside the boundaries of the provided map. The system behaves like a global localizer for urban environments, without using an actual GPS. Experimental results show the performance of the proposed method in large scale urban environments using route network description (RNDF) segment-based maps.
Accurate localization is a fundamental task in order to achieve high levels of autonomy in robot navigation and robustness in vehicle positioning. Localization systems often depend on GPS due to its affordability and convenience. However, it is well known that GPS is not fully reliable, since satellite positioning is not available anytime, anywhere. This is the case of extreme scenarios such as underwater or underground navigation, for instance. In the context of urban navigation, GPS signals are often affected by buildings ('urban canyon' effect) blocking the reception or generating undesirable jumps due to multi-path effects. Fortunately, the use of a priori maps can help in the localization process.
There are maps already available for certain environments, such as the digital maps used for road positioning. Moreover, accurate maps can be built using GIS tools for many environments, not only urban, but also off-road settings, mining areas, and others.
Usually the vehicle position is evaluated by combining absolute information such as GPS with onboard sensors such as encoders and IMUs. Since GPS information is not permanently available, significant work has been carried out in order to integrate external sensors such as laser and sonar in the localization process such as in (Leonard & Durrant-Whyte, 1991), (Guivant & Nebot, 2001).
A priori information, such as digital maps, has been used to obtain accurate global localization, usually fusing information into Bayesian filters (Fox et al., 2001). Maps of the

environment are used in (Dellaert et al., 1999) in order to differentiate between obstacles and free space, and to bias the distribution of particles. In (Oh et al., 2004) map-based priors are incorporated into the motion model, and in (Liao et al., 2003) the localization is performed using particles constrained to a Voronoi map. Segment-based maps are often used in driving assistance systems for urban vehicle navigation. In this context, maps are usually defined as graphs that represent the road network connectivity. (Najjar et al., 2005) proposes a method that uses segment-based maps and Kalman filtering to perform an accurate localization constrained to the map. The scheme relies on the selection of the appropriate candidate segment from the dataset, considering multiple criteria and the estimated location of the vehicle. A similar approach has been presented in (Taylor & Blewitt, 2000).

In this paper we present a method to perform global localization in urban environments using segment-based maps in combination with particle filters. The contributions of the proposed architecture are two. Firstly, the likelihood function is generated as a grid, based on the segment-based map. In this way the scheme can efficiently assign weights to the particles in real time, with minimum memory requirements and without any additional pre-filtering procedure. Multiple hypotheses are handled transparently by the filter. The second contribution is an extension to the observation model, called local history-based observation. Hereby, the filter is able to deal with 'out-of-map' navigation cases, a feature that is highly desirable since the map can be incomplete or the vehicle can be actually located outside the boundaries of the map.

This paper is organized as follows: Bayesian methods are briefly introduced in Section **2**, with particular emphasis on localization using particle filters. Section **3** describes our proposed approach to perform vehicle localization using particle filters and route network description (RNDF) segment-based *a priori* maps. The likelihood generation scheme is shown and an extension to the observation model, the local history-based observation, is introduced. Results illustrating the performance of the system in experiments undertaken in a large urban environment are provided and detailed in Section **4**. Conclusions and future work are finally discussed in Section **5**.

2. Bayesian Localization

Bayesian methods (Arulampalam et al., 2002) provide a rigorous general framework for dynamic state estimation problems. Bayesian approaches aim at building the probability density function (PDF) of the state vector based on all the available information. Vehicle localization can be understood as a Bayesian estimation problem. If the robot's location at time *k* is expressed as the vector

$$X_k = \left[x_k, y_k, \phi_k\right]^T , \tag{1}$$

then the localization problem implies the recursive estimation of the PDF

$$p_{X_k | z^{(k)}}\left(X_k \mid z^{(k)}\right) \tag{2}$$

where $z^{(k)}$ is the sequence of all the available sensor measurements until time *k*.

In the rest of this paper we will express the probability functions without the sub indices, i.e. any expression such as $p(\xi)$ will mean $p_{\xi}(\xi)$.

If we suppose that the posterior $p\left(X_{k-1} \mid z^{(k-1)}\right)$ at time $k-1$ is available, then the prior at time *k* (due to a prediction step) is:

$$p\left(X_k \mid z^{(k-1)}\right) = \int p\left(X_k \mid X_{k-1}\right) \cdot p\left(X_{k-1} \mid z^{(k-1)}\right) \cdot dX_{k-1} \tag{3}$$

where $p\left(X_k \mid X_{k-1}\right)$ is the process model for the system, i.e. the motion model of the vehicle. At time *k* a set of measurements z_k become available allowing the synthesis of a posterior that is obtained through a Bayesian update stage as:

$$p\left(X_k \mid z^{(k)}\right) = C \cdot p\left(X_k \mid z^{(k-1)}\right) \cdot p\left(z_k \mid X_k\right) \tag{4}$$

where the constant C is a normalization factor, and $p\left(z_k \mid X_k\right)$ is the likelihood function for the related observation $z = z_k$ at state X_k, i.e. it is the observation model.

For the linear Gaussian estimation problem, the required PDF remains Gaussian on every iteration of the filter. Kalman Filter relations propagate and update the mean and covariance of the distribution. For a nonlinear non-Gaussian problem, there is in general no analytic expression for the required PDF. A particle filter can estimate parameters with non-Gaussian and potentially multimodal probability density functions. The PDF is represented as a set of random samples with associated weights, and the estimates are computed based on these samples and weights.

For a set of *N* particles at time k, denoted as $\left\{X_k^i, w_k^i\right\}_{i=1}^{N}$, the approximated posterior is:

$$p\left(X_k \mid z^{(k)}\right) \approx \sum_{i=1}^{N} w_k^i \cdot \delta\left(X_k^i - X_k\right) \tag{5}$$

where $\delta(z-a)$ is the Dirac delta function which is ∞ if $z = a$ and zero otherwise. The weights are normalized such that $\sum_{i=1}^{N} w_k^i = 1$ to guarantee $\int p\left(X_k \mid z^{(k)}\right) \cdot dX_k = 1$.

Using the approximated posterior, the prior at time *k* (after a prediction stage) becomes:

$$\begin{aligned} p\left(X_k \mid z^{(k-1)}\right) &= \int p\left(X_k \mid X_{k-1}\right) \cdot p\left(X_{k-1} \mid z^{(k-1)}\right) \cdot dX_{k-1} \approx \\ &\approx \int p\left(X_k \mid X_{k-1}\right) \cdot \sum_{i=1}^{N} w_{k-1}^i \cdot \delta\left(X_{k-1}^i - X_{k-1}\right) \cdot dX_{k-1} = \\ &= \sum_{i=1}^{N} w_{k-1}^i \cdot \int p\left(X_k \mid X_{k-1}\right) \cdot \delta\left(X_{k-1}^i - X_{k-1}\right) \cdot dX_{k-1} = \\ &= \sum_{i=1}^{N} w_{k-1}^i \cdot p\left(X_k \mid X_{k-1} = X_{k-1}^i\right) \end{aligned} \tag{6}$$

$$\Downarrow$$

$$p\left(X_k \mid z^{(k-1)}\right) \approx \sum_{i=1}^{N} w_{k-1}^i \cdot p\left(X_k \mid X_{k-1} = X_{k-1}^i\right) \tag{7}$$

where $p\left(X_k \mid X_{k-1} = X_{k-1}^i\right)$ is the process model applied to each sample X_{k-1}^i.

This density is then used as the proposal distribution $q\left(X_k\right)$ from which samples are drawn, i.e. $X_k^i \sim q\left(X_k\right)$.

The update equation for the weights is:

$$w_k^i = w_{k-1}^i \cdot \frac{p\left(z_k \mid X_k^i\right) \cdot p\left(X_k^i \mid X_{k-1}^i\right)}{q\left(X_k^i\right)} \tag{8}$$

The posterior is again approximated as:

$$p\left(X_k \mid z^{(k)}\right) \approx \sum_{i=1}^{N} w_k^i \cdot \delta\left(X_k^i - X_k\right) \tag{9}$$

There are several details and issues before expression (8) is achieved, however those are not discussed in this paper. Several remarkable papers discuss details about the Monte Carlo approach applied to localization problems, e.g. (Dellaert et al., 1999).

3. Constrained Localization in Segment-Based Maps

The sole use of GPS can be insufficient to obtain an accurate estimation of the vehicle's location in urban navigation, as can be seen in the experiments presented in Figure 1. Figure 1a shows the trajectory reported by the GPS in an urban environment in Sydney, Australia. GPS inconsistencies such as jumps and discontinuities are shown in the close-up image in Figure 1b. In both images GPS points are shown with blue crosses, while the cyan dashed lines indicate the reported sequence and clearly show the undesirable behavior.

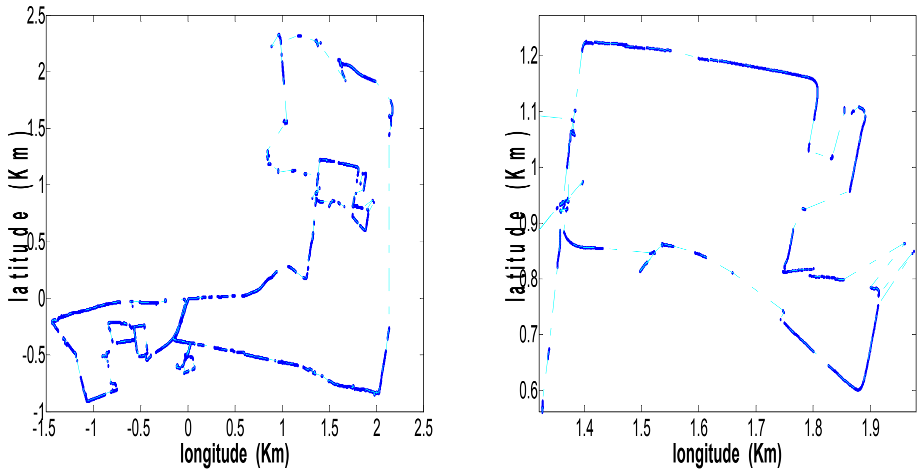


Fig. 1. GPS samples acquired during a trip in a urban context. GPS was not available in many parts of the test. The blue points show the GPS measurements and the cyan broken line indicates the sequence of measurements. Cyan segments (without superimposed blue points) mean that there was an interruption in the GPS measurements or that there was a sudden discontinuity in the measured positions.

In this section we present an approach to perform global localization in urban environments using segment-based maps in combination with particle filters. First we formulate a likelihood function using segment-based maps, such as the route network description file (RNDF). From now on we will base our formulation on a map, defined as a segment-based map. We show how the proposed scheme can be used to efficiently assign weights to the particles without any particular segment evaluation or candidate pre-selection procedure, and how multiple hypotheses can be handled automatically by the localization filter. We then introduce an extension to the observation model in order to deal with 'out-of-map' localization.

3.1. Likelihood Generation

In the context of this paper there are two definitions of likelihood functions; those are the *Base Likelihood* and the *Path Likelihood*. The Base Likelihood is intended to model the likelihood of a point, i.e. the likelihood of that point being located on a valid road. The second definition, the Path Likelihood, is the likelihood of a pose (position and heading) and some associated path to it, of being located on a valid road. Both likelihood definitions are based on the road map.

3.1.1. Road Map Definition

The road map behaves as a permanent observation model, i.e. it defines a constraint that, in combination with a dead-reckoning process, makes the pose states observable. The route network description file (RNDF) is, in the context of the Darpa Urban Challenge (Darpa, 2006), a topological map that provides *a priori* information about the urban environment. It includes GPS coordinates for the location of road segments, waypoints, stop signs and checkpoints, as well as lane widths. Fig. 2 shows a sample of a road map.

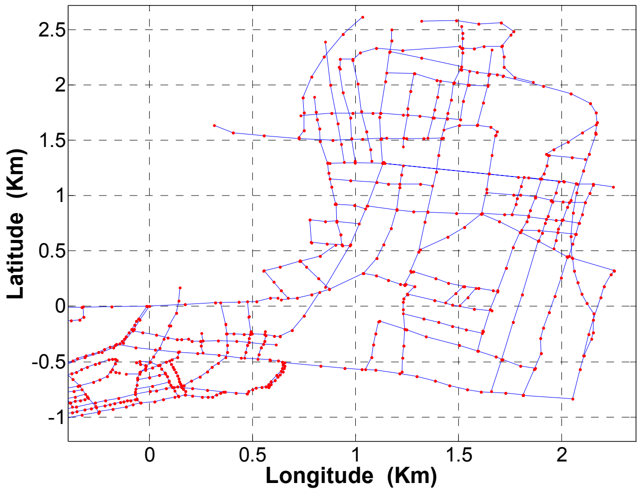


Fig. 2 Road map defined by a set of segments (continuous blue lines). The segments are defined by points expressed in a global coordinate frame (red points).

One of the key ideas in the presented approach is the synthesis of a local grid representation of the segment based map to compute the likelihood function. The advantage of this local grid-based formulation is that it can efficiently generate the likelihood function for the particles in real time and with minimal memory requirements. It can also select the possible roads (segments in the RNDF map) that can be used to perform the observation for each of the particles, without additional high-level evaluation of the potential candidate segments.

3.1.2 Base Likelihood

For a set of N particles at time k, $\left\{X_k^i, w_k^i\right\}_{i=1}^N$ we calculate the likelihood $p\left(z_k \mid X_k^i\right)$ based on a given segment based map as:

$$p\left(map \mid X\right) = \max_{j=1}^{N}\left\{f\left(X, S_j, C_j\right)\right\} \tag{10}$$

where $\left\{S_j\right\}_{j=1}^N$ is the set of all segments that define the road map (centers of roads) and C_j denotes the properties of segment S_j (width, lanes, lane directions etc.). The function $f(.)$ is function of the distance of the position component of the state X with respect to the center of the segment S_j. Function $f(.)$ is also dependent on each segment's properties such as its width and directions of its lanes.
A simplified version of (10) is

$$p(map \mid X) = \begin{cases} 1; & if \quad X \in L_{map}(RNDF, \Omega_k) \\ 0; & if \quad X \notin L_{map}(RNDF, \Omega_k) \end{cases} \tag{11}$$

where the region Ω_k is just a convex hull that contains all the particles $\{X_k^j\}_{j=1}^N$.

The region $L_{map}(RNDF, \Omega_k)$ defines the roads as bands using the segment's locations and their widths provided in the RNDF. This computation is performed only in the area covered by the particles, i.e. Ω_k. Through the local computation of small windows of interest, the likelihood function can be evaluated for all of the particles in real time.

Another advantageous capability of this local grid-based formulation for the likelihood function is in terms of multi-hypothesis handling. The localization filter can deal with multi-hypothesis cases without any additional procedure in this regard. Since the selection of the local window is spanned by the area covered by the set of current particles, Ω_k, the filter can inherently perform observations for all the hypothetical vehicle locations.

3.1.3. Path Likelihood

If the observation model is directly implemented by just applying the base likelihood function on the current instances of particles, then the system will not be able to deal with *out-of-map* localization cases. The term out-of-map means that the vehicle is allowed to travel through unknown sections of the map (i.e. sections not included in the *a priori* road map).

Since the particles will be biased by areas of high base likelihood, the population will tend to cluster towards those regions that are assumed to be consistent with the map (e.g. existing roads on the map). This can be a desirable behavior if we assume the map is complete and the vehicle remains on it at all times. However, if we want to cope with non-existent roads, detours or other unexpected situations not considered in the RNDF representation, then this policy might lead to an inconsistent localization as the applied likelihood is not consistent with the reality.

In general the convergence of the localization filter can be improved by considering the recent history of the particles within a certain horizon of past time.

Definition of the Associated Paths

An ideal procedure would be to have particles to represent the current pose and the path of the vehicle. By matching that path with the map it would be possible to evaluate the likelihood of that hypothesis.

One way to maintain a path would be by augmenting the state vector with delayed versions of the current state. However this increase in the dimensionality of the estimated state (already a 3 DoF one) would imply a high increase in the needed number of particles.

Alternatively, there is a highly efficient way to synthesize and associate a path for each particle at time *k*. This path is obtained through combination of a dead-reckoning estimation (i.e. obtained from an independent estimation process) and the current value of the particle X_k^i. This associated trajectory is realistic for short horizons of time as the dead reckoning prediction is valid just for a short term. Given a particle $X_k^i = \begin{bmatrix} x_k^i & y_k^i & \phi_k^i \end{bmatrix}^T$ we can apply dead-reckoning estimation in reverse in order to synthesize a hypothetical trajectory $\xi^i(t')$, $t' \in [k-\tau, k]$, where the value τ defines some horizon of time. This trajectory ends exactly at the particle instance, i.e. $\xi^i(k) = X_k^i$.

The estimated dead-reckoning trajectory is usually defined in a different coordinate system as it is the result of an independent process. The important aspect of the dead-reckoning estimate is that its path has good quality in relative terms, i.e. locally. Its shape is, after proper rotation and translation, similar to the real path of the vehicle expressed in a different coordinate frame.

If the dead-reckoning estimate is expressed as the path $\mu^i(t') = (x_\mu(t'), y_\mu(t'), \phi_\mu(t'))$ then the process to associate it to an individual particle and to express it in the global coordinate frame is performed according to:

$$\begin{gathered}(x_\xi(t'), y_\xi(t')) = (x_k^i, y_k^i) + R_{(\Delta_\phi(k))} \cdot (\Delta_x(t'), \Delta_y(t')) \\ \phi_\xi(t') = \phi_k^i + \phi_\mu(t') - \phi_\mu(k)\end{gathered} \tag{12}$$

where: $(\Delta_x(t'), \Delta_y(t')) = (x_\mu(t'), y_\mu(t')) - (x_\mu(k), y_\mu(k))$, $\Delta_\phi(k) = \phi_k^i - \phi_\mu(k)$ and R_α is a rotation matrix for a yaw angle α .The angle ϕ_k^i is the heading of the particle X_k^i and $\phi_\mu(k)$ is the heading of the dead-reckoning path at time *k*.

Clearly, at time $t' = k$ the difference $(\Delta_x(t'), \Delta_y(t'))$ must be $(0,0)$, and $\phi_\mu(t')\big|_{t'=k} - \phi_\mu(k) = 0$, consequently $\xi^i(t')\big|_{t'=k} = X_k^i$.

Fig. 3 (left) shows a dead-reckoning path and how it would be used to define the associated paths of two hypothetical particles. The associated paths Fig. 3 (right) are just versions of the original path adequately translated and rotated to match the particles' positions and headings.

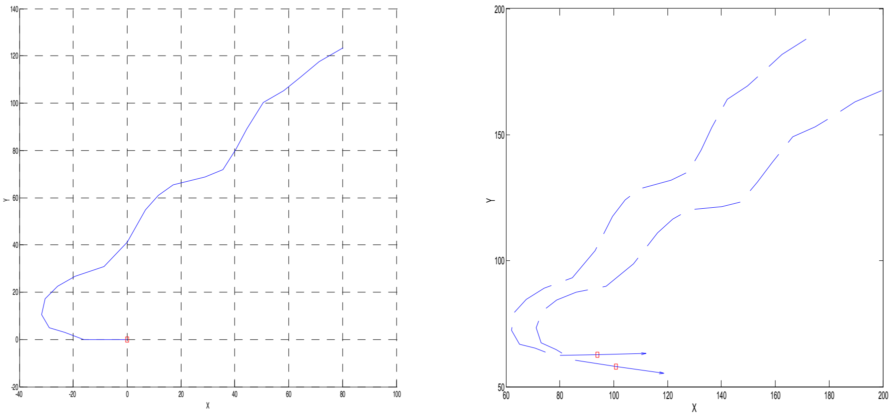


Fig. 3. The left picture shows a dead-reckoning path, expressed in a coordinate frame defined by the position and heading of the last point (red square). The right picture shows the same path associated to two arbitrary particles, expressed in a common coordinate frame

The new likelihood of a particle is now evaluated through the likelihood of its associated path with respect to the road map:

$$p^{*}\left(z_k \mid X_k^i\right) = \int_{k-\tau}^{k} p\left(z_k \mid \xi^i\left(t'\right)\right) \cdot dt', \tag{13}$$

where $p\left(z_k \mid \xi\right)$ is the base likelihood of the point ξ, i.e. likelihood of point ξ being on the RNDF map (as defined in (11)).
In order to avoid the effect of time scale (i.e. speed) on the path likelihood, we focus the evaluation of the likelihood on the intrinsic parameter of the path, integrating over the path in space and not in time:

$$p^{*}\left(z_k \mid X_k^i\right) = \int_{0}^{l_s} p\left(z_k \mid \xi^i\left[s\right]\right) \cdot ds, \tag{14}$$

where $\xi^i\left[s\right]$ is the path expressed in function of its intrinsic parameter s and l_s is the length of integration over the path. The integration of the hypothetical path can be well approximated by a discrete summation

$$p^{*}\left(z_k \mid X_k^i\right) = \sum_{j=1}^{N_J} p\left(z_k \mid \xi^i\left[s_j\right]\right) \tag{15}$$

where the samples of the intrinsic parameter s_j are homogeneously spaced (although that is not strictly relevant).
Some additional refinements can be considered for the definition of (13), for instance by considering the direction of the road. This means that the base likelihood would not be just a function of the position, it would depend on the heading at the points of the path . A path's segment that crosses a road would add to the likelihood if where it invades the road it has a consistent direction (e.g. not a perpendicular one).
Fig. 4 shows an example of a base likelihood (shown as a grayscale image) and particles that represent the pose of the vehicle and their associated paths (in cyan). The particles' positions and headings are represented blue arrows. The red arrow and the red path correspond to one of most likely hypotheses.
By applying observations that consider the hypothetical past path of the particle, the out-of-map problem is mitigated (although not solved completely) for transition situations. The transition between being on the known map and going completely out of it (i.e. current pose and recent path are out of the map) can be performed safely by considering an approach based on hysteresis.
The approach is summarized as follows: If the maximum individual path likelihood (the likelihood of the particle with maximum likelihood) is higher than K_H then the process keeps all particles with likelihood $\geq K_L$. These thresholds are defined by $100\% > K_H > K_L > 0\%$. If the maximum likelihood is $< K_H$ then the process keeps all the particles and continues the processing in pure prediction mode. Usual values for these thresholds are $K_H = 70\%, K_L = 60\%$.

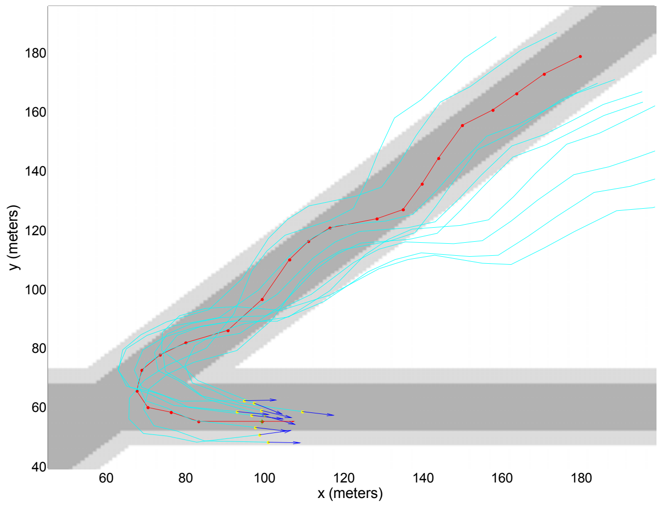


Fig. 4. A synthetic example. This region of interest (ROI) is a rectangle of 200 meters by 200 meters. A set of particles and their associated paths are superimposed to an image of base likelihood.

In the synthetic example shown in Fig. 4 the region of interest (ROI) is a rectangle of 200 meters by 200 meters. This ROI is big enough to contain the current population of particles and their associated paths.
Although all the particles are located on the road (high base likelihood); many of their associated paths abandon the zones of high base likelihood. The most likely particles are those that have a path mostly contained in the nominal zones. It can be seen the remarkable effect of a wrong heading that can rotate the associated path and make it to abandon the zones of high base likelihood (i.e. the road sections in gray).
Some particles have current values that escape the dark gray region (high base likelihood zones) however their associated paths are mostly contained in the roads. That means the real vehicle could be actually abandoning the road. This situation is repeated in Fig. 5 as well, where all the particles are located outside of the nominal road although many of them have paths that match the map constraints.
When the filter infers that the vehicle has been outside the map for sufficient time (i.e. no particles show relevant part of their paths consistent with the map), no updates are performed on the particles, i.e. the filter works in pure prediction mode.
When the vehicle enters the known map and eventually there are some particles that achieve the required path likelihood, i.e. higher than K_H, then the filter will start to apply the updates on the particles.
However this synchronization is not immediate. There could be some delay until some associated paths are consistent with the map -- the fact that a particle is well inside the road does not mean that its likelihood is high. It needs a relevant fraction of its associated path history to match the road map in order to be considered “inside the map”.

This policy clearly immunizes the filter from bias when incorrect particles are temporarily on valid roads.

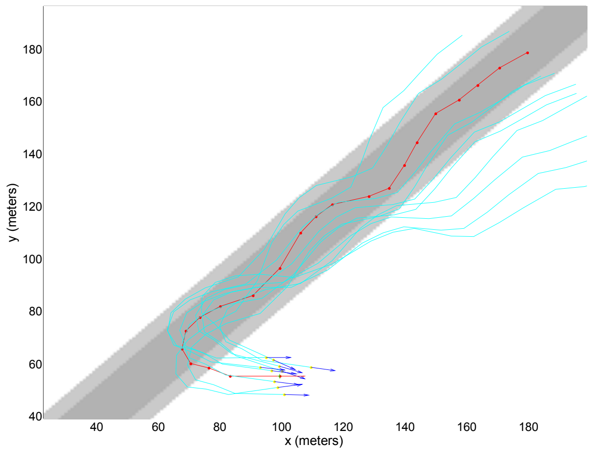


Fig. 5. This can be the situation where a vehicle temporarily abandons the road. It can be seen that although all the particles would have low base likelihood many of them have high likelihood when their associated paths are considered. Particles outside the road (low Base Likelihood) but having a correct heading would have high Path Likelihood.

4. Experimental Results

Long term experiments have been performed in urban areas of Sydney. The road maps were created by an ad-hoc Matlab tool that allowed users to define segments on top of a satellite image obtained from Google Earth. These road maps were low quality representations of the roads. This disregard for the quality of the definition of the road maps was done on purpose with the goal of exposing the approach to realistic and difficult conditions. Fig. 7 and Fig. 8 show the road map used in the estimation process. Fig. 2 shows part of the used road map as well.

The dead-reckoning process was based on the fusion of speed and heading rate measurements. The heading rate was provided by low cost three dimensional gyroscopes. A diversity of additional sensors were available in the platform (PAATV/UTE project) although those were not used in the estimation process and results presented in this paper. All the experiments and realistic simulations have validated the satisfactory performance of the approach.

Figures 7, 8 and 9 present the position estimates as result of the estimation process. Those are shown in red (Figure 7) or in yellow (Figures 8 and 9) and are superimposed on the road map. In some parts of the test the vehicle went temporarily outside the known map. Although there was not a predefined map on those sections it was possible to infer that the estimator performed adequately. From the satellite image and the over-imposed estimated path, a human can realize that the estimated path is actually on a road not defined in the *a priori* map (Fig. 9).

It is difficult to define a true path in order to compare it with the estimated solution. This is because the estimator is intended to provide permanent global localization with a quality usually similar to a GPS. Figures 10, 11 and 12 present the estimated positions and corresponding GPS estimates although those were frequently affected by multipath and other problems.

5. Conclusions and Future Work

This paper presented a method to perform global localization in urban environments using segment-based maps together with particle filters. In the proposed approach the likelihood function is locally generated as a grid derived from segment-based maps. The scheme can efficiently assign weights to the particles in real time, with minimum memory requirements and without any additional pre-filtering procedures. Multi-hypothesis cases are handled transparently by the filter. A path-based observation model is developed as an extension to consistently deal with out-of-map navigation cases. This feature is highly desirable since the map can be incomplete, or the vehicle can be actually located outside the boundaries of the provided map.

The system behaves like a virtual GPS, providing accurate global localization without using an actual GPS.

Experimental results have shown that the proposed architecture works robustly in urban environments using segment-based road maps. These particular maps provide road network connectivity in the context of the Darpa Urban Challenge. However, the proposed architecture is general and can be used with any kind of segment-based or topological *a priori* map.

The filter is able to provide consistent localization, for extended periods of time and long traversed courses, using only rough dead-reckoning input (affected by considerably drift), and the RNDF map.

The system performs robustly in a variety of circumstances, including extreme situations such as tunnels, where a GPS-based positioning would not render any solution at all.

The continuation of this work involves different lines of research and development. One of them is the implementation of this approach as a robust and reliable module ready to be used as a localization resource by other systems. However this process should be flexible enough to allow the integration with other sources of observations such as biased compass measurements and even sporadic GPS measurements.

Other necessary and interesting lines are related to the initialization of the estimation process, particularly for cases where the robot starts at a completely unknown position. Defining a huge local area for the definition of the likelihood (and spreading a population of particles in it) is not feasible in real-time. We are investigating efficient and practical solutions for that issue.

Another area of relevance is the application of larger paths in the evaluation of the Path Likelihood. In the current implementation we consider a deterministic path, i.e. we exploit the fact that for short paths the dead-reckoning presents low uncertainty to the degree of allowing us to consider the recent path as a deterministic entity. In order to extend the path validity we need to model the path in a stochastic way, i.e. by a PDF. Although this concept is mathematically easy to define and understand it implies considerable additional computational cost.

Finally, the observability of the estimation process can be increased by considering additional sources of observation such the detection of road intersections. These additional observations would improve the observability of the process particularly when the vehicle does not perform turning maneuvers for long periods.

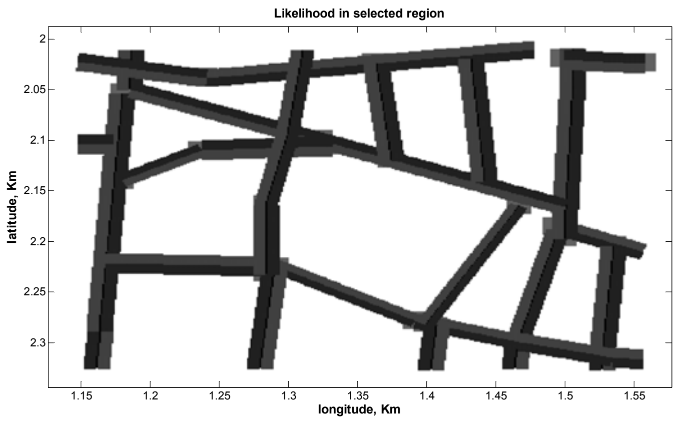


Fig. 6. A local base likelihood automatically created. This ROI is defined to be the smallest rectangle that contains the hypothetical histories of all the current particles. The different colors mean different lanes although that property was not used in the definition of the base likelihood for the experiment presented in this paper.

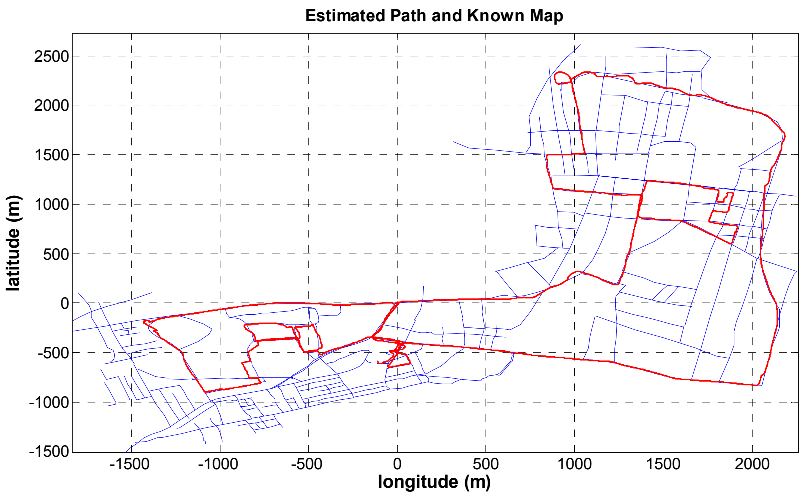


Fig. 7. Field test through Sydney. The system never gets lost. Even feeding the real-time system lower quality measurements (by playing back data corrupted with additional noise) and removing sections of roads from the a-priori map) the results are satisfactory. The red lines are the obtained solution for a long trip.

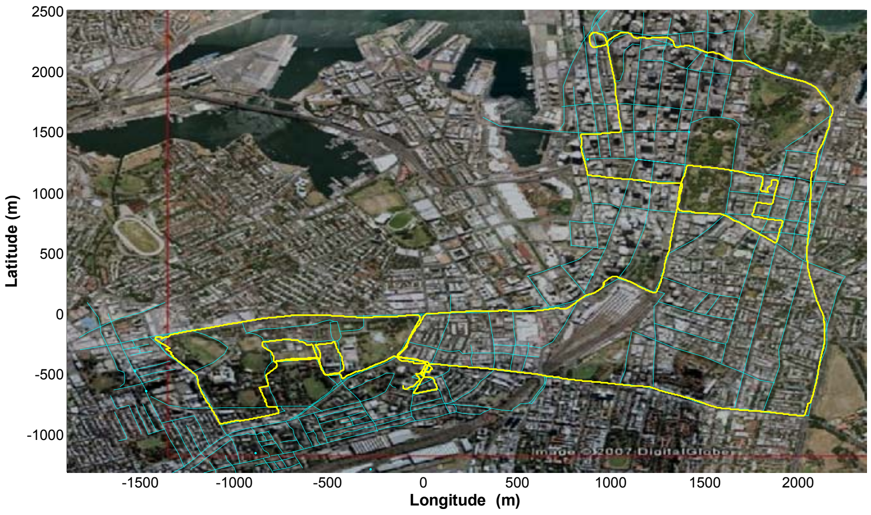


Fig. 8. Estimated path (in yellow) for one of the experiments. The known map (cyan) and a satellite image of the region are included in the picture.

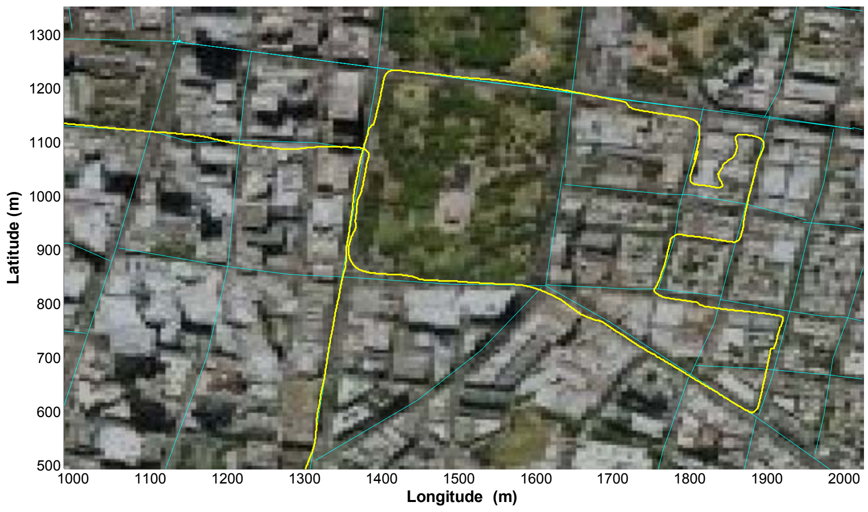


Fig. 9. A section of Fig. 8 where the solution is consistent even where the map is incomplete (approximately x=1850m, y=1100m).

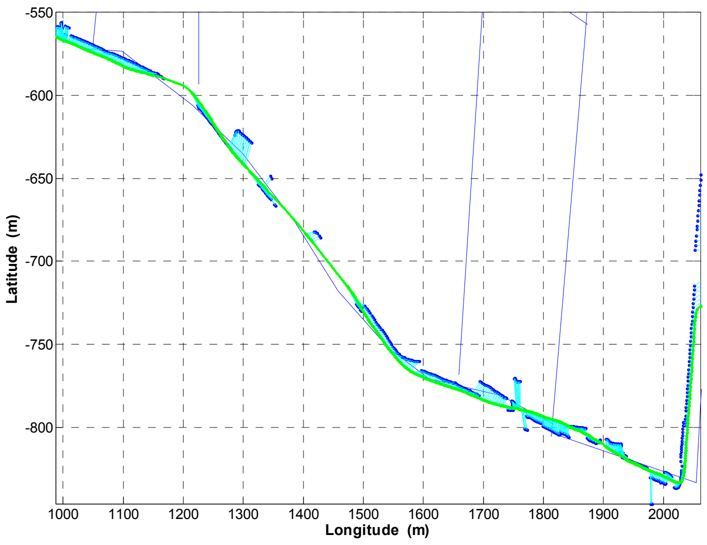


Fig. 10. A comparison between the estimated solution and the available GPS measurements. The green dots are the estimated solution and the blue ones correspond to GPS measurements. The segments in cyan connect samples of GPS and their corresponding estimated positions (i.e. exactly for the same sample time). The blue lines are the map's segments.

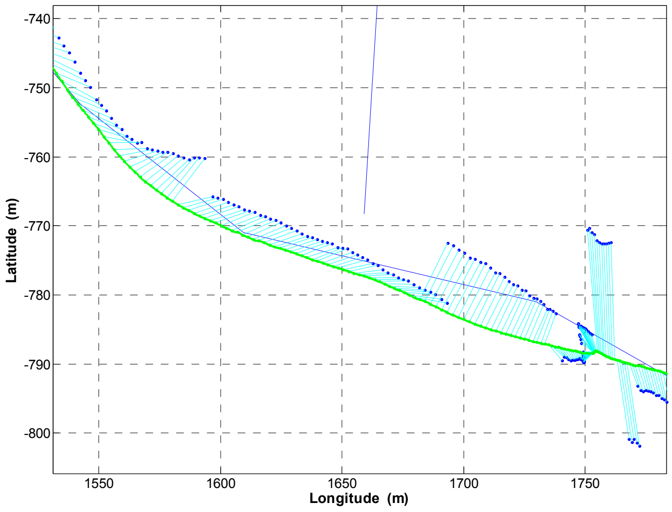


Fig. 11. A detailed view of Figure 10. It is clear that the GPS' measurements present jumps and other inconsistencies.

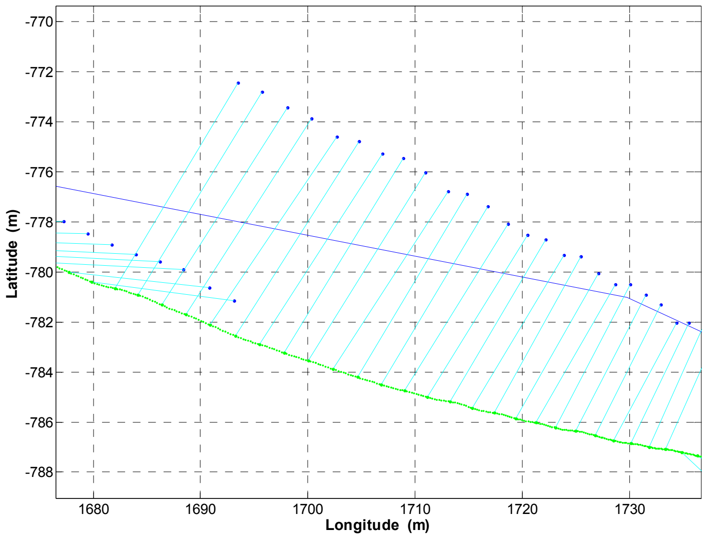


Fig. 12. A close inspection shows interesting details. The estimates are provided at frequencies higher than the GPS (5Hz). The GPS presents jumps and the road segment appears as a continuous piece-wise line (in blue), both sources of information are unreliable if used individually.

6. References

S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A Tutorial on Particle Filters for On-line Non-linear/Non-Gaussian Bayesian Tracking," *IEEE Transactions on Signal Processing,* vol. 50, no. 2, pp. 174–188, 2002.

F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo Localization for Mobile Robots," in *International Conference on Robotics and Automation*. Detroit: IEEE, 1999.

D. Fox, S. Thrun, W. Burgard, and F. Dellaert, "Particle Filters for Mobile Robot Localization," in *Sequential Monte Carlo Methods in Practice*, A. Doucet, N. de Freitas, and Gordon, Eds. New York: Springer, 2001.

J. E. Guivant and E. M. Nebot, "Optimization of the Simultaneous Localization and Map-building Algorithm for Real time Implementation," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, 2001, pp. 242–257.

J. Guivant, and R. Katz, "Global urban localization based on road maps," in *RSJ International Conference on Intelligent Robots and Systems, IROS*. San Diego, CA : IEEE, Oct. 2007, pp 1079-1084. ISBN: 978-1-4244-0912-9.

J. J. Leonard and H. F. Durrant-Whyte, "Simultaneous Map Building and Localization for an Autonomous Mobile Robot," *IEEE/RSJ International Workshop on Intelligent Robots and Systems*. IEEE, 1991.

L. Liao, D. Fox, J. Hightower, H. Kautz, and D. Schultz, “Voronoi Tracking: Location Estimation Using Sparse and Noisy Sensor Data,” in IEEE/RSJ *International Workshop on Intelligent Robots and Systems*. Las Vegas, USA: IEEE, 2003.

M. E. E. Najjar and P. Bonnifait, “A Road-Matching Method for Precise Vehicle Localization Using Belief Theory and Kalman Filtering,” *Autonomous Robots*, vol. 19, no. 2, 2005, pp. 173–191.

S. M. Oh, S. Tariq, B. N. Walker, and F. Dellaert, “Map based Priors for Localization,” in *IEEE/RSJ International Workshop on Intelligent Robots and Systems*. Sendai, Japan: IEEE, 2004.

G. Taylor and G. Blewitt, “Road Reduction Filtering using GPS,” in *3rd AGILE Conference on Geographic Information Science*, Helsinki, Finland, 2000.

ACFR, The University of Sydney and LCR, Universidad Nacional del Sur, “PAATV/UTE Projects,” Sydney, Australia, 2006.

Defence Advanced Research Projects Agency (DARPA), “DARPA Urban Challenge,” http://www.darpa.mil/grandchallenge/, 2006.

Google, “Google Earth,” http://earth.google.com/, 2007.

Object Localization using Stereo Vision

Sai Krishna Vuppala
Institute of Automation, University of Bremen
Germany

1. Introduction

Computer vision is the science and technology of machines that see. The theoretical explanation about the optics of the eye and the information about the existence of images formed at the rear of the eye ball are provided by Kepler and Scheiner respectively in the 16th century (Lin, 2002). The field of computer vision is started emerging in the second half of 19th century, since then researchers have been trying to develop the methods and systems aiming at imitating the biological vision process and therefore increasing the machine intelligence for many possible applications in the real world. The theoretical knowledge behind the perception of 3D real world using multiple vision systems has been published in various literature, and (Hartley & Zisserman 2000) (Klette at al., 1998) (Sterger , 2007) are few well known from them. As there are numerous applications of computer vision in service, industrial, surveillance, and surgical etc automation sectors, researchers have been publishing numerous methods and systems that address their specific goals. It is intended with most of these researchers that their developed methods and systems are enough general with respect to the aspects such as functional, robust, time effective and safety issues. Though practical limitations hinder the researchers and developers in achieving these goals, many are getting ahead providing the solutions to the known and predicted problems.
The field of computer vision is supporting the field of robotics with many vision based applications. In service robotics action interpretation and object manipulation are few examples with which the computer vision supports the humans. According to (Taylor & Kleeman, 2006), three things are almost certain about universal service robots of the future: many will have manipulators (and probably legs!), most will have cameras, and almost all will be called upon to grasp, lift, carry, stack or otherwise manipulate objects in our environment. Visual perception and coordination in support of robotic grasping is thus a vital area of research for the progress of universal service robots. Service robotic tasks are usually specified at a supervisory level with reference to general actions and classes of objects. An example of such a task would be: Please get 'the bottle' from 'the fridge'. Here, getting the bottle is the intended action, 'bottle' belongs to the class of objects that are manipulated, and 'fridge' belongs to the class of objects in/on which the objects to be manipulated lie. The content of the manuscript addresses the object manipulation tasks using stereo vision for applications of service robotics. Motivation of the content of the manuscript stems from the needs of service robotic systems FRIEND II and FRIEND III

(Functional Robot with dexterous arm and user-frIENdly interface for Disabled people) that are being developed at IAT (Institute of Automation, University of Bremen, Germany). The systems FRIEND II and FRIEND III are shown in figure 1 a) and b). The objects of interest to be manipulated are shown in figure 2.

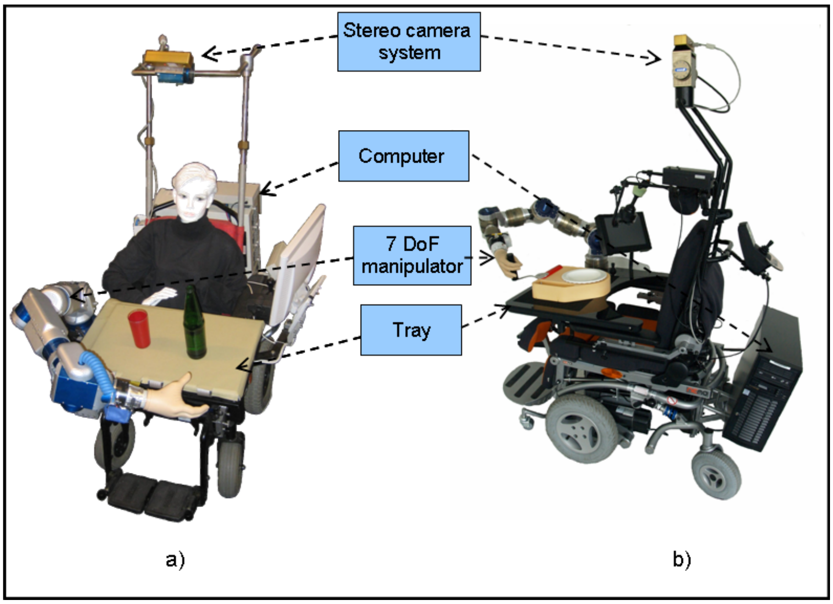


Fig. 1. Rehabilitation robotic systems a) FRIEND II b) FRIEND III

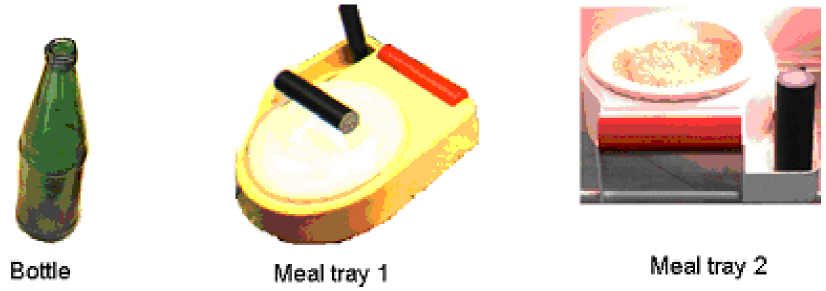


Fig. 2. The objects of interest for the manipulation purpose

In order robot manipulators to perform their actions autonomously, 3D environment information is necessary. In order to reconstruct 3D object information using cameras various approaches have been investigated since few decades. According to (Lange at al., 2006), the 3D object information recovery techniques are broadly classified into passive and active sensing methods. Passive sensing methods require relatively low power compared to the active methods. Passive sensing methods are similar to the biological vision process. The characteristic contrast-related features in images (i.e. cues such as shading, texture) of the observed scene are used in extracting the 3D information. In active sensing methods compared with passive methods high accuracy measurements can be obtained. Active sensing relies on projecting energy into the environment and interpreting the modified sensory view. CMOS time-of-flight cameras can capture complete 3D image in a single measurement cycle (Lange, 2000), however such a technology is limited with timing resolutions and, the possibility of the 3D information perception depends on the object surface properties.

Considering passive sensing methods, though any number of cameras can be used in 3D vision process, the methods based on stereo vision plays optimal role considering the functional, hardware, and computational issues. Visual servoing system (Corke, 1996) (Hutchinson, 1996) come into this this category. The two major classes of such systems are position- and image based visual servoing systems. In position based visual servoing systems, they are further classified into closed- and open loop systems. Depending on the requirements of the strategy, they can be realized using single, or two cameras. Closed loop systems have high accuracy since the overall system error is minimized using the feedback information, where as in an open-loop system the error depends on the performance of the system in a single step. In contrast to the closed approaches look-then-move is an open loop approach for the autonomous task executions; the vision system identifies the target location in 3D and the robot is driven to the location of interest. In look-then-move approach, the accuracy of the overall robotic system depends on the camera calibration accuracy and the manipulator accuracy in reaching the specified locations in the real world (Garric & Devy 1995). In a system like FRIEND II look-then-move approach is prefered since object manipulation, collision avoidance, and path planning are planned using a standlone stereo camera unit.

Stereo triangulation is the core process in stereo based 3D vision applications, in order to calculate 3D point stereo correspondences information is used. Precisely identified stereo correspondences give precise 3D reconstruction results. Any uncertainty in the result of stereo correspondences can potentially yield a virtually unbounded uncertainty in the result of 3D reconstruction (Lin, 2002). Precise 3D reconstruction additionally requires well calibrated stereo cameras as the calibrated parameters of the stereo view geometry are used in the stereo triangulation process (Hartley & Zissermann 2000). In addition to that 3D reconstruction accuracy further depends on the length of base line (Gilbert at al., 2006), if the base line is shorter the matching process is facilitated, and if the base line is larger the 3D reconstruction accuracy is higher. Approaches for finding the stereo correspondences are broadly classified into two types (Klette at al., 1998); they are intensity and feature-based matching techniques. The state of the art stereo correspondence search algorithms mostly based on various scene and geometry based constraints. These algorithms can not be used for the texture less objects as the reconstructed obejct information is error prone.

The pose (position and orientation) of an object can be estimated using the information from a single or multiple camera views. Assuming that the internal camera parameters are known, the problem of finding the object pose is nothing but finding the orientation and position of object with respect to the camera. The problem of finding the pose of the object using the image and object point correspondences in the literature is described as a perspective n-point problem. The standard perspective n-point problem can be solved using systems of linear equations if correspondences between at least six image and scene points are known. Several researchers provided solutions for this problem considering at least 3 object points. According to Shakunaga in his publication on pose estimation using single camera (Shakunaga, 1991) described that an n-vector body with n >=3 gives at most 8 rotation candidates from object image correspondences. In case if n < 3, the recovery of the rotation of n-vector body is not possible.

The content of the manuscript discusses selection of stereo feature correspondences and determining the stereo correspondences for opted features on texture less objects in sections 2 and 3 respectively; section 4 presents tracking object pose using 2 object points. ; section 5 presents 3D object reconstruction results; Conclusion and References are followed in sections 6 and 7 respectively.

2. Selection of Stereo Feature Correspondences

Scene independent 3D object reconstruction requires information from at least two camera views. Considering stereo vision rather than multiple vision systems for this purpose eases the computational complexity of the system. Stereo vision based 3D reconstruction methods require stereo correspondence information. Traditional problems in finding the stereo correspondence are occlusion, regularity/ repetitiveness. Traditionally these problems are solved using intensity and feature based stereo matching techniques. However, absolute homogeneous surfaces without any sharp features (i.e. edges, corners etc) do not provide proper stereo correspondence information. The domestic objects to be manipulated in the FRIEND environment are some examples of such kind. As the considered object (either each part or the whole body) has uniform color information the intensity based correspondence search methods often provide improper stereo correspondence information. Therefoere, alternatively the edges of the green bottle are observed for the stereo correspondence analysis. Out of all possible edges of green bottle that is shown in figure 3, only the orientation edges of the bottle can be considered for the 3D bottle reconstruction; this is because the stereo correspondence information between other kinds of edges is typically lost.

2.1 Resolutions between 3D and 2D

The 3D reconstruction accuracy depends on, the accuracy of calibration parameters, the sub pixel accuracy of stereo correspondences and the length of the baseline. In addition to these factors, the 3D reconstruction accuracy further depends on the pixel size, the focal length of the camera, and the distance between the camera and the object. In the following the feasible spatial resolution of object surface in the projected scene is analyzed. Figure 4 illustrates the projection of the object surface on to a single image row. The feasible geometrical resolution of the object surface projected on to one row of the image plane is calculated using formula (1). Similarly the feasible resolution of the object projection on to the column of the object plane can be calculated. The width and the height of the pictured scene are also can be approximately calculated multiplying the number of pixels along the rows and columns multiplied with respective feasible resolutions.

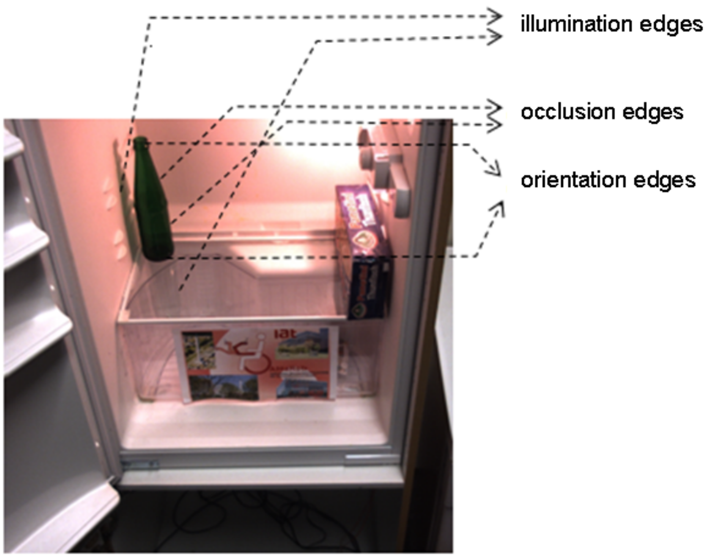


Fig. 3. Various possible edges of green bottle from FRIEND Environment

$$feasible\ resolution = \frac{pd}{f} \tag{1}$$

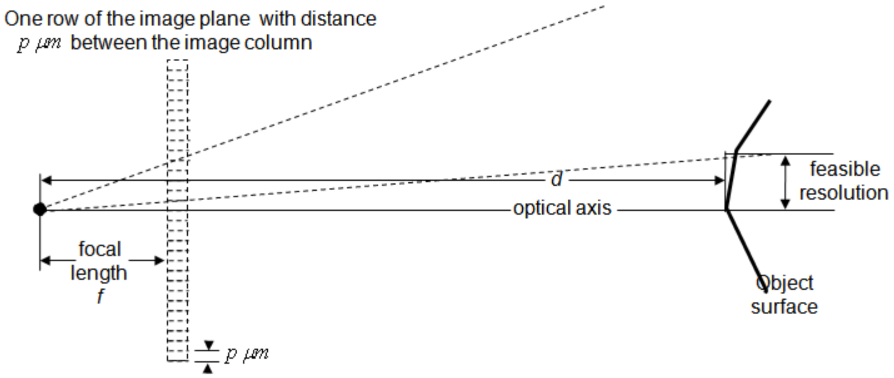

One row of the image plane with distance
p μm between the image column
d
optical axis
feasible resolution
focal length f
Object surface
p μm

Fig. 4. Feasible spatial resolution on object surface in the projected scene (Klette at al., 1998)

E.g: The size of the SONY CCD Sensor ICX204AK that is used in the bumblebee stereo vision system has the pixel size of $4.54\mu m \times 4.54\mu m$, the focal length of the camera is $4mm$, and the assumed situation of the object is about 1m far from the camera. The feasible resolution of the camera along the row is $1.125mm$/column and the feasible resolution along the column is $1.125mm$/row. Similarly with a higher focal length camera such as $6mm$ the feasible resolution becomes $0.75mm$/column. Implicitely, with decrease in feasible resolution increases the 3D reconstruction accuracy. From equation (1), the feasible resolution is proportional to the distance between the camera and the object, as the distance between the object and the camera decreases the feasible resolution for the object decreases and therefore increases the accuracy of 3D reconstruction processes.

2.2 Stereo Feature Correspondences for Homogeneous Object Surfaces

For a pair of matching line segments in stereo images, any point on the first line segment can correspond to every other point on the second line segment, and this ambiguity can only be resolved if the end-points of the two line segments are known exactly. Consider that the line segment AB shown in figure 5 lies on a cylindrical object surface is identified in stereo images. The stereo correspondences for the end points *A* and *B* are considered to be available. As no other external object hides the object line, all the projected object points on line *AB* have corresponding stereo image pixels.

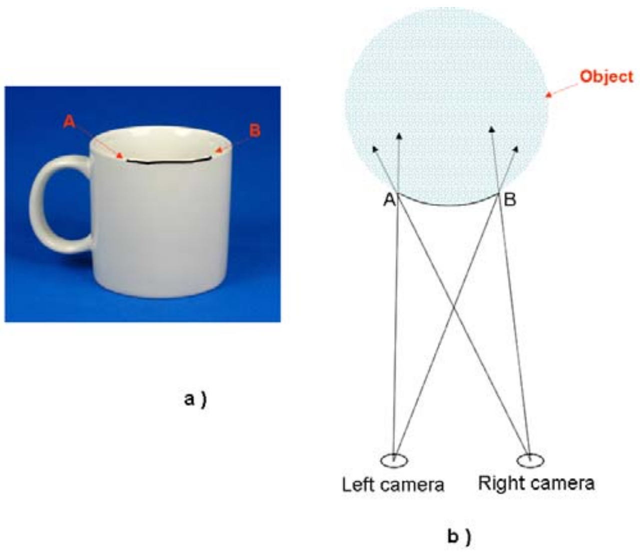
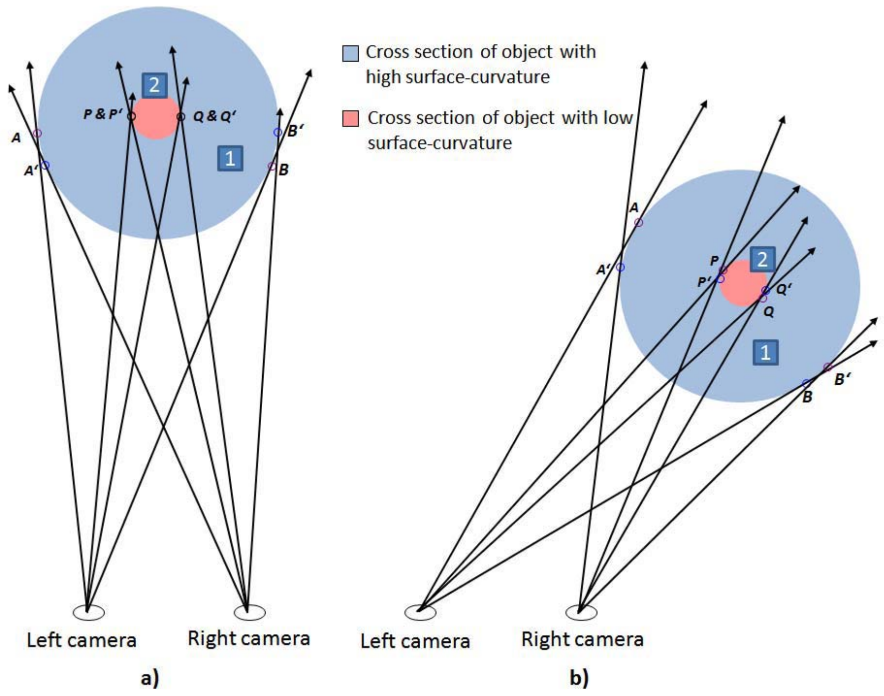


Fig. 5. a) Line segment on the object surface which could be identified in stereo images; b) Significant view of the line segment for the stereo correspondence analysis

It is not always possible to have (or detect) such line segments on the objects whose end points stereo correspondences could be determined. Therefore as an alternative, the points on the object contour (the edges of the object boundary that represent object shape in images) are considered for the stereo correspondence analysis. Surface curvature of an object is inversely proportional to the square of its radius. If the object surface curvature is not equal to zero the visible object contours in stereo images though may appear similar but do not correspond to the same object points. Figures 6a and 6b show the cross sections of object surfaces with different radius of curvatures observed from two different positions, object indexed with 1 has low surface curvature and object indexed with 2 has high radius of curvature. In figures 6a and 6b, the points *A* and *B* are any two points obtained from the contour image of the left camera for object indexed 1, and *A′* and *B′* are the similar points obtained for object indexed 1 from the contour image of the right camera; similarly, points *P* and *Q* are any two points obtained from the contour image of the left camera for object indexed 2, and P' and Q' are similar points obtained for object indexed 2 from the contour image of the right camera. Though (A,A'), (B,B'), (P,P') and (Q,Q') do not give real correspondences, an approximation of the reconstructed information is not far from the real 3D values. Also, intuitively one can say that the selected points from the high surface curvature give more appropriate stereo correspondences. Therefore if objects do not provide intensity or feature based stereo correspondence information, geometric specific feature points on object surfaces with larger radius of curvatures can be considered in order to have less 3D reconstruction errors. As the considered domestic objects within the FRIEND environment are texture less, approach discussed as above has been used to reconstruct the objects. Figure 7 shows the selection of contour regions in order to have less stereo correspondence errors for the objects that are intended to be manipulated in the FRIEND environment (bottle and meal tray). The contour area marked in the Fig 7a has the high radius of curvature on the surface of the green bottle. In case of the meal tray, as we have considered to reconstruct the meal tray using the red handle (shown in Fig 7b), the marked contour area of the meal tray has the high radius of curvature. Therefore in order to reconstruct these objects feature points are extracted from the marked contour object regions.



🟦 Cross section of object with high surface-curvature
🟥 Cross section of object with low surface-curvature
A
A'
P & P'
Q & Q'
B'
B
1
2
P
P'
Q'
Q
Left camera
Right camera
a)
b)

Fig. 6. object surfaces with different radius of curvatures from two different views

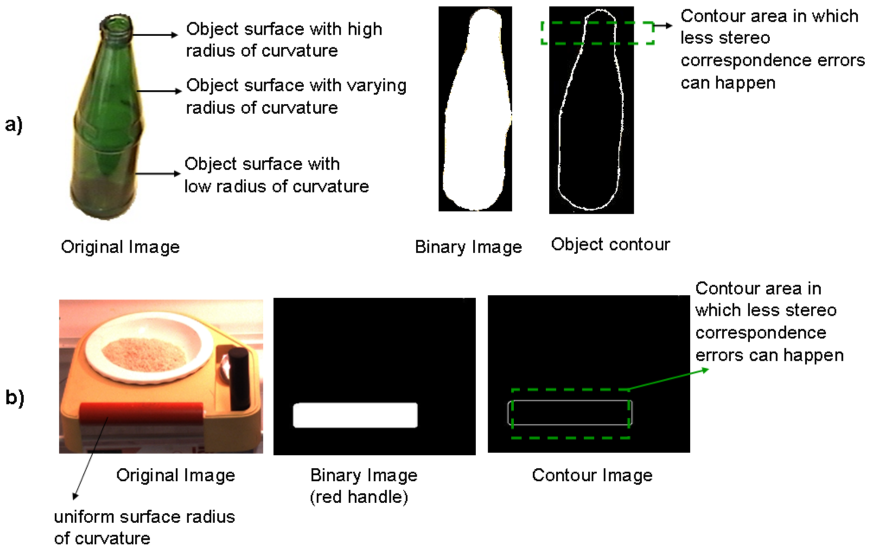

a)
Object surface with high radius of curvature
Object surface with varying radius of curvature
Object surface with low radius of curvature
Contour area in which less stereo correspondence errors can happen
Original Image
Binary Image
Object contour
b)
Contour area in which less stereo correspondence errors can happen
Original Image
Binary Image (red handle)
Contour Image
uniform surface radius of curvature

Fig. 7. Images of a bottle and meal tray in different image processing phases

3. Determining the stereo correspondences

As there are foreseen uncertainties exist in the selection of stereo feature correspondences, various constraints have been investigated which give object feature points those could be used for the 3D reconstruction of the object.

3.1 Homography based Stereo Correspondence Calculation

The homography calculated between the stereo images for an intended real world plane can be used for finding the stereo correspondence information. Such a calculated homography between the stereo images must only be used for finding the stereo correspondences for the 3D points that lie on the real world plane considered for the calculation of the homography, otherwise a plane induced parallax phenomena occurs. In order to calculate the homography of the 3D plane at least four stereo correspondences from the stereo image pair are needed (Hartley & Zisserman 2000).

Figure 8 illustrates the plane induced parallax phenomena. The homography induced by the 3D plane π in the stereo images is sufficient for the stereo correspondence calculation of 3D points $U_{\pi 1}$ and $U_{\pi 2}$ those lie on plane π. Whereas, the stereo correspondence for the 3D point U_3 which does not lie on plane π cannot be calculated using H. This is because the mapping $H : u_2 \mapsto u_2'$ is a valid mapping for the 3D point Uπ2 as it lies on plane π, consequently the mapping H always gives u2′ for u2. Therefore the supposed stereo correspondences for the 3D point U_3, i.e. the mapping $H : u_2 \mapsto u_3'$ cannot be done.

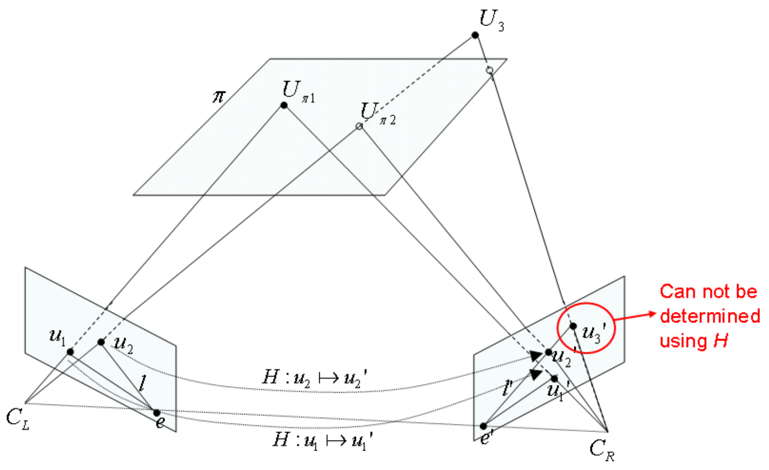


Fig. 8. Sketch: Illustration of plane induced parallax phenomena

The practical results for the plane induced parallax can be clearly seen from figure 9. The homography matrix *H* between the stereo image pair has been calculated for the chess board π from a real world scene. Fig 9a shows the selected test points on the left image and fig 9b shows the calculated correspondences based on the computed homography. The stereo correspondences for the points u_1', u_2', and u_3' on chess board are computed correctly, the stereo correspondences for the points u_4' and u_5' which are not in the chess board plane are wrongly computed. Therefore it is difficult to use such an approach for the 3D object reconstruction.

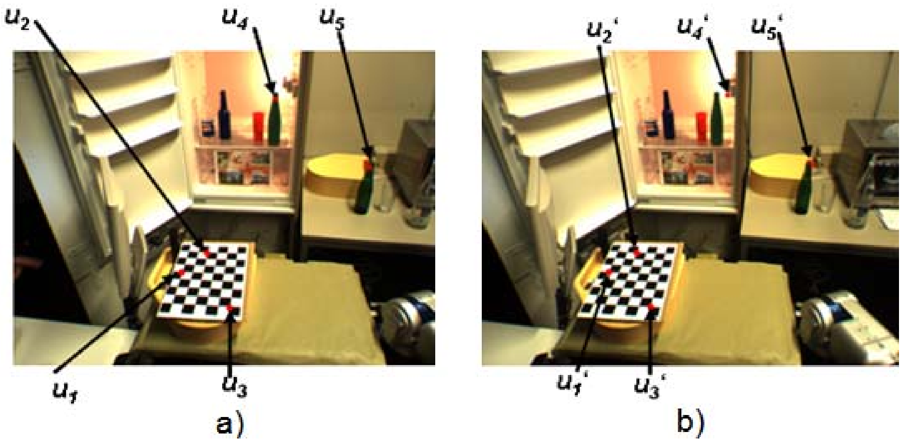


Fig. 9. Calculated stereo correspondences based on homography

3.2 Stereo Correspondences via Disparity Map

In order to find the stereo correspondence information, the algorithm proposed by Birchfield and Tomasi (Birchfield & Tomasi, 1998) has been used. The stereo correspondence

information is obtained using the disparity information obtained from the rectified stereo images. The algorithm improves the previous work in the area of stereo matching. Usually it is difficult to match the pixels of both images when there are big areas without any pattern, this algorithm can handle large untextured regions so the produced disparity maps become reliable. It uses dynamic programming with pruning the bad nodes, which reduces lot of computing time. The image sampling problem is overcome by using a measure of pixel dissimilarity that is insensitive to image sampling. The selection of the parameters of the algorithm is fundamental to obtain a good disparity map. However, the calculated pixel dissimilarity is piecewise constant for textureless objects.
Figures 10, 11, and 12 show the stereo images for 3 different bottle positions and the respective disparity maps. Implementation of the algorithm proposed by the authors of (Birchfield & Tomasi, 1998) is available as open source with OpenCV library. The parameters required for the algorithm are determined offline. In order to test the usability of this algorithm in current context where objects of interest are patternless and are needed to be reconstructed in 3D, a point on the middle of the bottle neck is manually selected on the left image, and the respective right correspondence information is calculated using the disparity value. The stereo correspondences are marked with red circles in the figures; the centres of the circles are the actual positions of the stereo correspondences. In figures 10 and 12, the calculated right correspondence information is approximately in correct position for a human eye, where as in figure 11, the calculated right correspondence information is clearly shifted by several pixels.

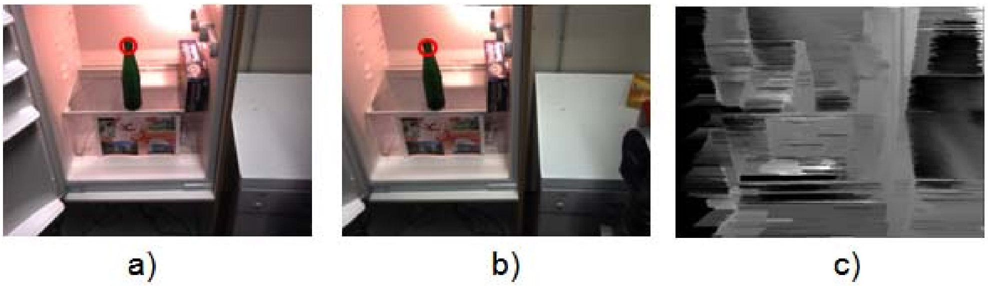


Fig. 10. Disparity map for bottle position 1; stereo images a) Left b) Right; c) Disparity map

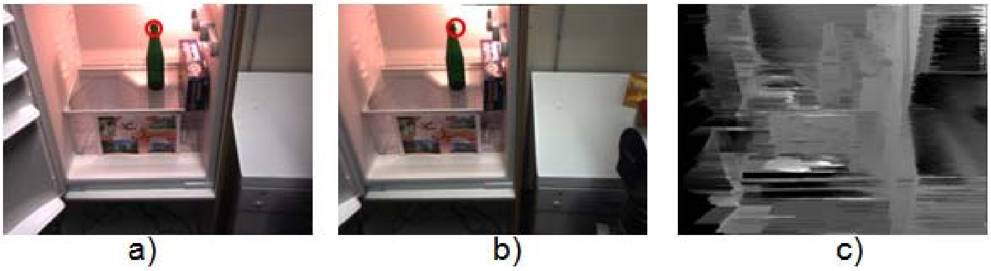


Fig. 11. Disparity map for bottle position 2; stereo images a) Left b) Right; c) Disparity map

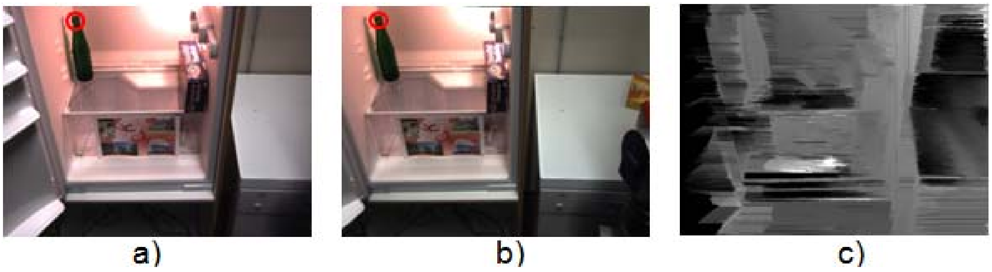


Fig. 12. Disparity map for bottle position 3; stereo images a) Left b) Right; c) Disparity map

The computation time required for the calculation of the disparity map (1024 * 768 pixels) is approximately 3s on a Pentium computer with 3 GHz processor. As the calculated pixel dissimilarity is piecewise constant for a textureless object, object such as green bottle shown in these figures possess only single, or few gray values, which means the obtained stereo correspondence information is clearly error prone. The quality of disparity map depends also on the selection of the optimal parameters. The calculated stereo correspondence values are presented in section 3.3.2 along with the values obtained using the appoaches discussed in section 3.3.1 and 3.3.2.

3.3 Constrained approaches for 3D point reconstruction

If we know the stereo correspondence information for a 3D object point to be calculated we will have 4 linear independent equations and 3 unknown 3D coordinates of the object point, and therefore we will have an over determined solution to solve the 3D unknown coordinates. In case we do not know anyone of the stereo pair (i.e. either right or left) we will have an under determined solution, i.e 5 unknowns to be solved with 4 equations; or ignoring the equations related with right stereo correspondence result in 3 unknowns to be solved with two equations. Therefore, if we know one of the coordinates from the 3D point (i.e. either *X*, *Y*, or *Z*) the rest coordinates can be calculated either using a single camera, or using a stereo camera system.

3.3.1 Calculation of 3D unknowns using single camera

Let *U*=(*X*, *Y*, *Z*, 1) is the homogeneous notation for a real world 3D object point to be calculated. Let *Z* coordinate is known as a priory (i.e. measured or calculated) from *U*. Let *u*=(*x*, *y*, 1) is the respective image point for *U*. Let *P* be the projection matrix (or camera matrix) that relate the image point and real world point for the considered camera. Then, the following cross product holds between the image point and the real world point (Tsai, 1987).

$$u \times PU = 0 \tag{2}$$

we therefore will have two linear independent equations

$$x(p^{3T}U) - (p^{1T}U) = 0 \tag{3}$$

$$y(p^{3T}U) - (p^{2T}U) = 0 \tag{4}$$

where, p_i, i = 1, 2, and 3 is a column vector that represents the ith row of the projection matrix. Using (3) we can have,

$$\Rightarrow X = K_6 + K_7 Y \tag{5}$$

where $K_7 = K_5 / K_4$, $K_6 = K_3 / K_4$, $K_5 = p_{12} - K_{32}$, $K_4 = K_{31} - p_{11}$, and $K_3 = K_2 - K_{33}$, $K_{31} = p_{31}x$, $K_{32} = p_{32x}$, and $K_{33} = xK_1$, $K_2 = p_{13}Z + p_{14}$,. $K_1 = p_{33}Z + p_{34}$. p_{ij} is an element of projection matrix P with an index of i^{th} row and j^{th} column, here i= 1 or 3 and j= 1 till 4. Using (4), we can have

$$\Rightarrow X = K_{15} + K_{14} Y \tag{6}$$

where $K_{15} = K_{13} / K_{11}$, $K_{13} = K_2 - K_{10}$, $K_{12} = p_{22} - K_9$, $K_{11} = K_8 - p_{21}$, $K_{10} = yK_1$, $K_9 = yp_{32}$, and $K_8 = yp_{31}$. Solving (5) and (6) we have the final expression for Y coordinate in (7).

$$Y = (K_6 - K_{15})/(K_{14} - K_7) \tag{7}$$

As we have Y, the final expression for X can be obtained either from (5) or (6).

3.3.2 Calculation of 3D unknowns using stereo cameras

As the stereo correspondences lie along epipolar lines, the following points are analysed

- For an interested 3D object point in the left image, the values of 3D coordinates X, Y and Z lie along the left projection ray and are unique values for changes in the selection of correspondence point along the right epipolar line
- For an interested 3D object point in the left image, the values of 3D coordinates X, Y and Z lie along the left projection ray are either monotonously increasing, or monotonously decreasing, for one directional changes in the selection of correspondence along the right epipolar line

From these two observations, intuitively one can determine the rest two unknowns if any of the three unknown object point coordinates is known as a priory. In order to consider such an approach a closed loop method is suggested in the following. Gradient descent method or steepest descent method is one of the well known unconstrained optimization algorithms used to find the local minimum of a function $f(x)$. Since the gradient descent method is simple and each of the iteration can be computed fast we have considered this approach for our application. The method starts at an initial state (x_0) in the considered minimization function and moves from a state to its next state (x_i to x_{i+1}) based on the step length (a), the descent direction determined by the gradient (g_i). Selecting a constant step size consumes high computational time in reaching the minimum and also may results in imprecise results, therefore a variable step size rule presented in (Rohn, 1992) is used in our approach. The considered closed loop system is shown in Figure 13. The Z coordinate of the 3D point $U(X,Y,Z)$ is considered to be known and serves as reference input r. The left image point u_L for U is considered to be available. The gradient descent controller computes the right correspondence value u_R along the right epipolar line. The gradient descent method acts on the error value between the reference input and the calculated Z coordinate until the error reaches the tolerance value.

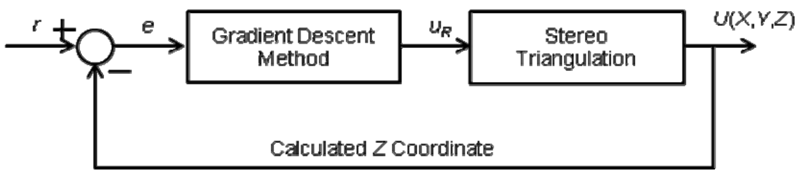


Fig. 13. Closed loop system for precise object localization using gradient descent method

Selected left image point on bottle (in pixels)	Right correpondence and 3D point calculation using method from section 3.2				3D point calculation using method from section 3.3.1				Right correpondence and 3D point calculation using method from section 3.3.2			
	Right pixel	X (m)	Y (m)	Z (m)	Right Correspondence	X (m)	Y (m)	Z (m) known	Right Correspondence	X (m)	Y (m)	Z (m) known
position #1: (412,140)	(280, 140)	0.543	0.543	0.530		0.525	0.541	0.538	(277.65, 140)	0.521	0.541	0.540
position #2: (495,84)	(389, 84)	0.842	0.471	0.442	Not Applicable	0.639	0.468	0.538	(370.424, 84)	0.635	0.468	0.540
Position #3: (239,52)	(120, 52)	0.715	0.727	0.534		0.707	0.726	0.538	(119.232, 52)	0.703	0.727	0.540

Table 1. Comparison of calculated stereo correspondence and 3D information using different approaches

Table 1 presents the values of selected left image points, calculated right correspondences and calculated 3D information for 3 different bottle positions from figures 10, 11 and 12. The calculated 3D information using the constrained methods descussed in section 3.3.1 and 3.3.2 are almost same (i.e. deviation of values is less than 5mm). Where as, the 3D values calucated using the method from section 3.2 for poistion2 are far from the values calcuated using the methods presented in sections 3.3.1 or 3.3.2, position1 has a deviation about 2cm in *X* and less than 5mm deviation in *Y* and Z directions, and position 3 values also are deviated less than 1cm. It is observed approach presented in section 3.2 is time consuming and strongly influenced with lighting conditions, texture on object surface, and back ground information etc; the constrained approaches presented in sections 3.3.1 and 3.3.2 have strong limitaion of knowing priory information.

4. Object Pose Estimation using Two Object Points

The algorithm proposed in this chapter supports the "look-then-move" approach in object manipulation tasks. In a "look-then-move" approach, it is required to have 6-Degrees of Freedom (DoF) of object pose in robot base coordinate system assuming the transformation between the robot base and the gripper coordinate system is known to the tolerable precision. The 6-DoF of object pose consists of 3-DoF for the position and the rest 3-DoF for the orientation of the object. The orientation and position of the camera in the object space are traditionally called the camera extrinsic parameters (Yuan, 1989) (Zhang, 2007). The problem of finding camera parameters is described as a perspective n-point problem, since the pose of the object is estimated based on the known set of object points that are identified in the image. Typically the problem is solved using the least squares techniques (Hartley &

Zisseman, 2000) (Lowe, 1987), where a large data set is needed in order to obtain a stable solution. Because of inherent computational complexity of such an approach it is often undesirable in the real-time robotic applications where the actions of the other components of the system depend on the stereo vision output. In order to overcome this problem certain number of researchers has tried to develop methods which require less number of object points for the pose estimation. A summary of the available solutions for pose estimation starting from 3 points is given in (Horaud, 1989). In (Yoon at al., 2003), object 3D pose is tracked using 3 reconstructed distinguishable blobs of an industrial object. The advantage of the algorithm proposed in this chapter is object pose can be tracked using only two object points, therefore it increases the possibility for estimating objects pose which have very less pattern information. Since the algorithm bases on the slope of a line in a plane, vector notation for these two points is not necessary. The constraints of the algorithm are:

- Transformation between the stereo camera system and the considered reference coordinate system is a priory known and kept constant throughout the algorithm execution.
- Objects are not allowed to rotate simultaneously about the multiple axes of the considered reference coordinate system.
- Objects are not allowed to rotate about the virtual line joining the considered two object points, or a parallel line to it in the reference coordinate system.

In a "look-then-move" approach the pose of the object with respect to the robot base has to be known. An example of such a robotic task which belongs to the field of service robotics is to 'fetch the object placed on the platform'. I.e. a compound transformation between the robot base and the ultimate object to be manipulated has to be known. In a next abstraction level the mentioned task can be divided further into two sub tasks, one is to find the pose of the platform with respect to the robot base and the other is to find the pose of the object with respect to the platform. In the first sub task the reference coordinate system belongs to the robot base and in the second sub task the reference coordinate system belongs to the platform. Satisfying the above mentioned constraints, the pose of the platform with respect to the robot base and the pose of the object with respect to the platform can be obtained using the proposed algorithm. In the following, section 4.1 explains the proposed algorithm for tracking the pose of an object, and section 4.2 presents the experimental results.

4.1 Tracking 3D Object Pose using Two Object Points

In this approach two object points are needed to track the object pose, once the object pose is tracked all the other required object points are reconstructed in a reference coordinate system using the priory knowledge of these points in object coordinate system. The algorithm execution is done in the following steps:

- In an initial object situation, the considered two object points are reconstructed in 3D of the reference coordinate system using the calibrated stereo cameras (Hartley & Zisseman, 2000).
- A virtual line joining the considered two 3D points makes an angle with each axis of the reference coordinate system. Using this line, the rotations of object around an axis of the reference coordinate system are tracked up to a span of 180° starting from the initial object situation in the reference coordinate system.

Figure 14 shows the initial situation of the object in 3D. The line joining the 3D points P_1 and P_2 is considered for the algorithm explanation. The orthographic projection of the line on the *xy*-plane forms a 2D line $P_1'P_2'$ which makes an angle *a* with *x*-axis. Similarly, the lines $P_1''P_2''$ and $P_1'''P_2'''$ formed in other planes make angles β and γ with *y* and *z* axes respectively.

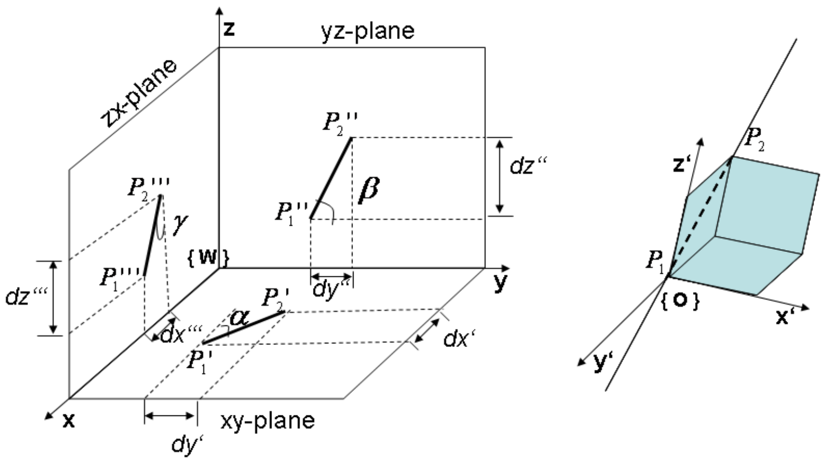


Fig. 14. Orthographic projections of 3D line on each plane of the reference coordinate system that correspond to an initial object situation

$$\alpha = \arctan(dy'/dx') \tag{8}$$
$$\beta = \arctan(dz''/dy'') \tag{9}$$
$$\gamma = \arctan(dx'''/dz''') \tag{10}$$

where dx' and dy' are the lengths of the line $P_1'P_2'$ on the *x* and *y* axes respectively. Similarly dy'' and dz'' are the lengths of the line $P_1''P_2''$ on the *y* and *z* axes, and dx''' and dz''' are the lengths of the line $P_1'''P_2'''$ on the *x* and *z* axes respectively. The ratios dy'/ dx', dz''/ dy'', and dx'''/dz''' are the slopes of the lines $P_1'P_2'$, $P_1''P_2''$, and $P_1'''P_2'''$ in *xy*, *yz*, and *zx* planes with respect to *x*, *y*, and *z* axes respectively.

4.1.1 Object Rotation around an Axis

When the object is rotated about an axis, the orthographic projections of considered 3D line of the object are changed in all the planes of the reference coordinate system. Figure 15 shows the orthographic projections of the considered object line on the 3 planes of the reference coordinate system when the object is rotated only about the *z*-axis of the reference coordinate system. In order to maintain the transparency, the point of rotation is shown at P_1 in Figure 15, the point of rotation can be any point in 3D. After the rotation, the new rotation angles are a^*, β^*, and γ^*. This information imlicitely tells us that the rotation of object around multiple axes of the reference coordinate system can not be tracked using the slope of the line in reference planes, because, when the object is rotated about multiple axes it has interlinked influence on rotation angles in all the planes.

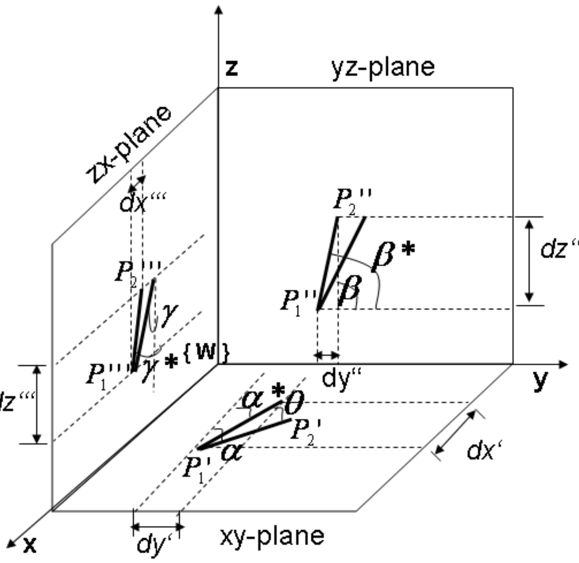


Fig. 15. Rotation of the object around z-axis of the reference coordinate system {*W*}

4.1.2 Calculation of rotation angle

Figure 16 shows the rotations of a line *l* in *xy*-plane. The lines l_i and l_i' are the instances of the line *l* for different slopes, where *i* = 1, 2, and 3. For each value of *i* the slope of any line l_i is m_i, and is same for the line l_i'. This information reveals that the slope of the line is same for 180° of its rotation in a plane. Relating this information for the rotations of object in 3D, using two object points object rotation angle around an axis of the reference coordinate system can be tracked uniquely up to a range of 180°. One can customize the range to be only positive or only negative or both positive and negative for object rotations from initial situation.

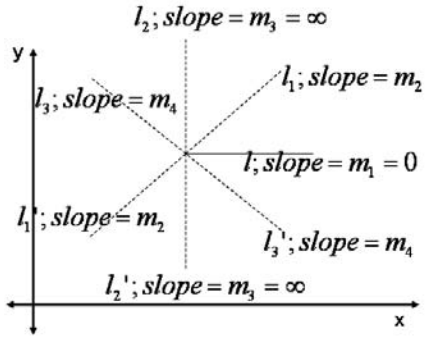


Fig. 16. Slope of the line '*l*' for its rotations in *xy*-plane

In the folllowing, we have considered the object rotation angle (θ) is in a range between -90° and +90° from the object initial situation. In fig 17, the line *l* is the orthographic projection of the considered 3D object line in the *xy*-plane, it makes an angle α with the *x*-axis in its initial object situation. Line *l** is the orthographic projection of the considered 3D object line when the object is rotated around the z-axis of the reference coordinate system for an angle *θ*. The line *l** makes an angle *a-Current* with the *x*-axis. The angle *θ* can be calculated using (11) and (12).

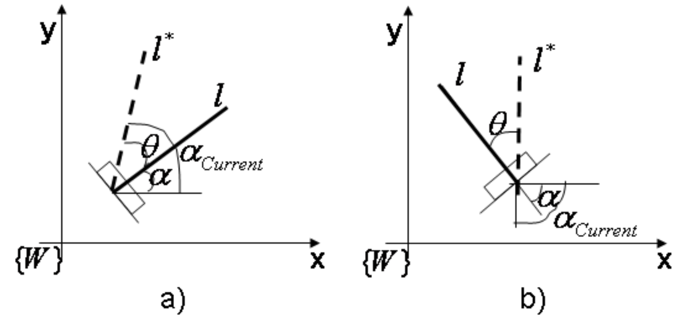


Fig. 17. Orthographic projection of 3D object line in the *xy*-plane; line *l* makes a) positive inclination (*a* >=0) and b) negative inclination (*a* < 0) in the *xy*-plane with *x*-axis

$$if\ \alpha \ge 0,\ \theta = \begin{cases} \alpha_{Current} - \alpha;\ if\ \alpha_{Current} > \alpha_{NegativeLimit} \\ \alpha_{Current} - \alpha + 180°;\ if\ \alpha_{Current} <= \alpha_{NegativeLimit} \end{cases} \tag{11}$$

$$if\ \alpha < 0,\ \theta = \begin{cases} \alpha_{Current} - \alpha - 180°;\ if\ \alpha_{Current} > \alpha_{PositiveLimit} \\ \alpha_{Current} - \alpha;\ if\ \alpha_{Current} <= \alpha_{PositiveLimit} \end{cases} \tag{12}$$

where, $\alpha_{NegativeLimit} = \alpha - 90°$ and, $\alpha_{PositiveLimit} = \alpha + 90°$

4.1.3 Position of object point in reference coordinate system

Figure 18 shows rotation of an object around *z*-axis of the reference coordinate system. The position of an object point *P*$_3$ is initially known in object coordinate system and is intended to be reconstructed in a reference coordinate system. According to the previous discussion, the line *l* joining the two object points *P*$_1$ and *P*$_2$ tracks the object rotation angle around *z*-axis up to a range of 180° from object's initial situation. When the object is rotated about *z*-axis of the reference coordinate system, the position of a point *P*$_3$ is estimated in the reference coordinate system using (13) and (14).

$$^{W}p_{P_3} = {}^{W}_{O'}T\ {}^{O'}p_{P_3} \tag{13}$$

$${}^{W}_{O'}T = \begin{bmatrix} rot(z,\theta) & {}^{W}p_{O'} \\ 0 & 1 \end{bmatrix} \tag{14}$$

where, *θ* = ∠(*l*, *l**) is obtained from stereo vision using (11) and (12). {*O*} and {*O'*} denote the object origins in initial situation and when the object is moved respectively. The coordinate system {*W*} denotes the reference coordinate system. $^{O'}p_{P_i}$ is same as $^{O}p_{P_i}$, and denotes the priory known position of the object point P_i , where i = 1,2, and 3, in object coordinate system. ${}^{W}p_{P_3}$ denotes the position of the point P_3 in the reference coordinate system. ${}^{W}p_{O'}$ is the position of the new object origin in the reference coordinate system, and in the considered case it is same as ${}^{W}p_{P_1}$. ${}^{W}p_{P_1}$ denotes the position of point P_1 in the reference coordinate system.

4.2 Experimental Results

In order to test the algorithm accuracy two small blobs as features are used on the meal tray. The proposed algorithm is tested for the rotations of the meal tray around z-axis of the reference coordinate system. The origin of the reference coordinate system is located about 90cm far from the middle point of the stereo vision system. The presented object rotations are ranged "from -80° till 80° in steps of 10°" around *z*-axis in the reference coordinate system from an initial object situation. Since object rotations are around the *z*-axis the rotation angle is tracked in *xy*-plane of the reference coordinate system. Figure 19 shows, calculated object rotation angle using the proposed algorithm versus the manually measured rotation angle, when the object is rotated about z-axis of the reference coordinate system. The Pearson correlation coefficient is a measure of extent to which two random variables are linearly related. The Pearson correlation coefficient for the calculated rotation angle and the manually measured rotation angle is 0.999, which means that they are strongly and positively correlated.

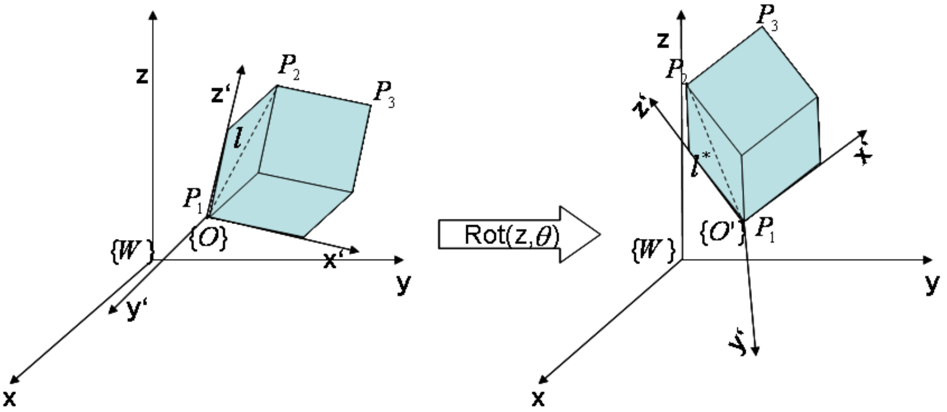


Fig. 18. Rotation of an object around z-axis of the reference coordinate system

Figure 20 shows the results for the reconstruction of an object point P_3 in the reference coordinate system, the position of the considered point in object coordinate system is priory known. Horizontal axis shows reconstructed 3D point P_3 using direct stereo correspondences. Vertical axis shows the calculated 3D point using the proposed algorithm with equations (13) and (14). The Pearson correlation coefficients for the x and y values of the calculated 3D point using direct stereo correspondences and proposed algorithm are 0.9992 and 0.9997 respectively, which clearly shows that they are strongly and positively correlated. Considering the Pearson correlation coefficient for the *z* value is not meaningful since it is always pointed to a constant value in the reference coordinate system for the considered case. Therefore the z-value is measured here with the standard deviation of the errors between the two methods, and is 0.005m.

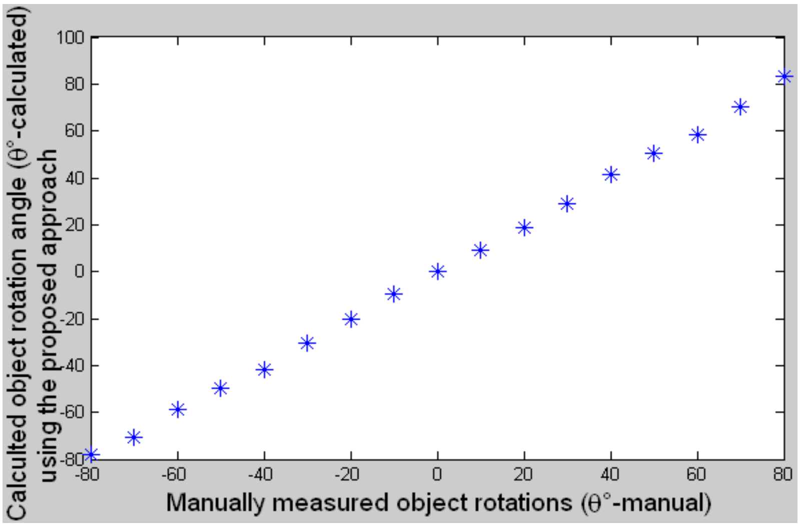


Fig. 19. Calculated rotation angle (θ°-calculated.) using proposed approach versus manually measured rotation angle (θ°-manual)

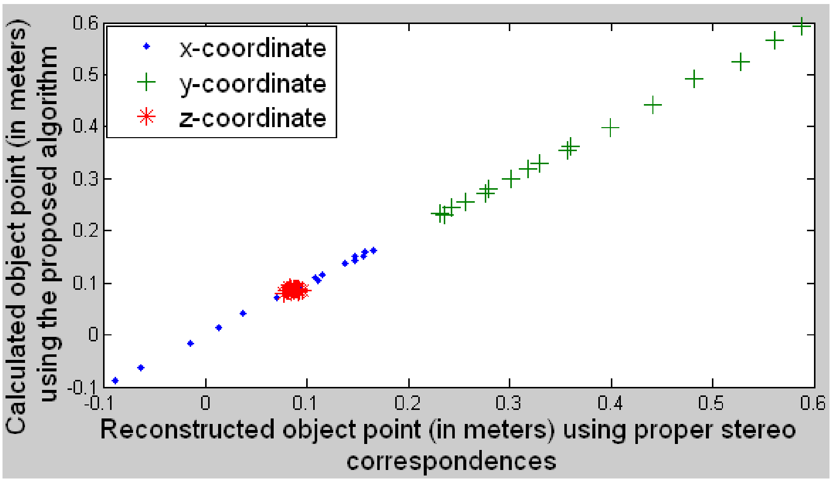


Fig. 20. Calculated 3D position of a point on the object using the proposed algorithm versus reconstructed position of the same point using direct stereo correspondences

5. 3D reconstruction results

In system FRIEND II stereo vision provides 3D object pose information in various scenarios and two of them are “serving drink” and “serving meal”. In serving drink scenario, bottle which is available in a refrigerator is fetched and the drink is poured into a glass that is placed on a smart tray, and finally the drink is served it to the user. In the serving meal scenario meal tray is taken from a refrigerator and is taken into a microwave, after getting meal warmed it is served to the user. In order to have synchronised communication among the various sensors of the robotic system the locations of the objects within the service robotic environment are described in a common coordinate system called world coordinate system. In both the mentioned scenarios objects of interest are needed to be precisely localized. In order to determine the transformations between stereo camera and big objects such as refrigerator and microwave special patterns (e.g. chessboard or AR-Marker) are used. Considering the coordinate systems of big objects as reference coordinate systems, meal tray and bottle poses in the FRIEND II environment are calculated. The 3D models of all objects in the FRIEND II environment are predefined and are used in path planning and collision avoidance tasks. In order to reconstruct the 3D object models (bottle and meal tray), feature points are selected from the surface regions that are suggested in section 2.2.

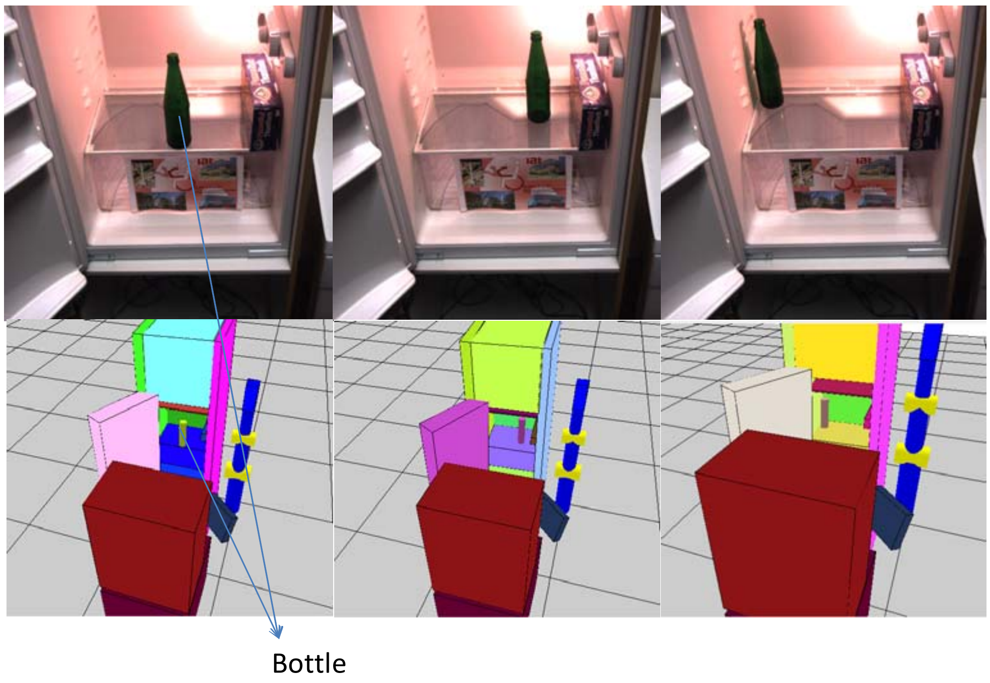


Fig. 21. Top row shows stereo left images of bottles inside a refrigerator; bottom row shows reconstructed bottle (cylindrical object) inside a refrigerator.

In the considered serving drink scenario it is expected that bottle always placed straight on a planar surface (i.e. either on a platform or on a shelf of the refrigerator). Therefore, in order to reconstruct the 3D model of the bottle only the position of bottle on the considered platform is required. Therefore in order to reconstruct the bottle position, the midpoint from the neck of the bottle is used as the feature point. The stereo correspondences are calculated using the method discussed in section 3.2. Three reconstructed positions of the bottles inside a refrigerator are shown in figure 21. The 3D model of the refrigerator is reconstructed based on the pattern information attached to the refrigerator; the 3D model of the robot arm is reconstructed using the position information provided by the encoder present in each joint. Small cylindrical object shown in the refrigerator model is the reconstructed 3D model of bottle. Numerous times the scenario has been successfully executed and is presented in international fairs ICORR 2007 at Noordwijk Holland, and in CeBIT 2008 at Hannover Germany. In case of no detection of the bottle or if the bottle presents outside the work area of the scenario execution regions, user is warned or informed accordingly. Due to the surface uniformity bottles used in FRIEND II could be grasped in any direction perpendicular to the principal axis.

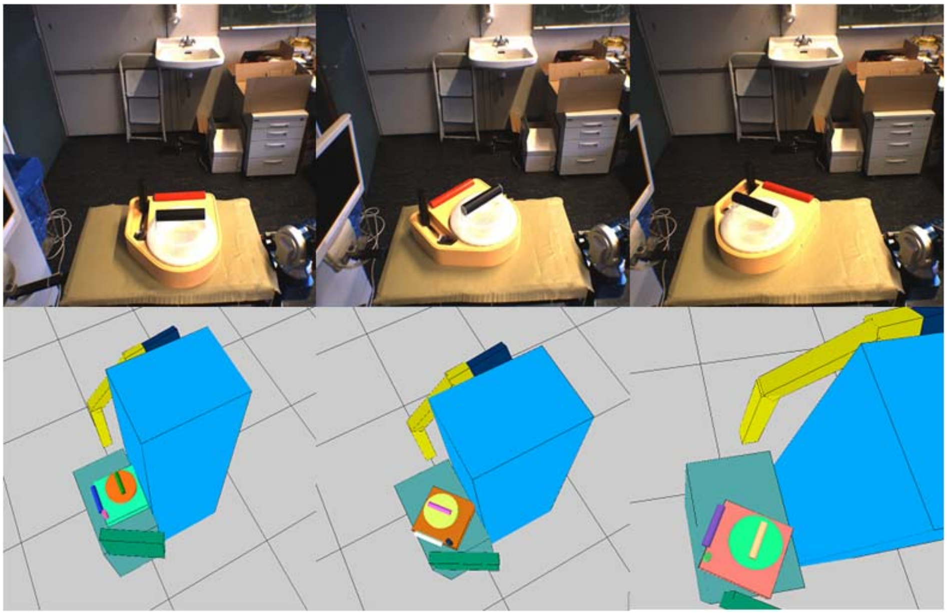


Fig. 22. Reconstructed meal tray poses from meal serving platform

Whereas meal trays used in FRIEND II environment could be grasped in only direction, i.e. a perpendicular direction to the principal axis of the meal tray handle. All components (spoon, meal cover, meal base, cover handle, meal plate) poses have to be precisely estimated in order robot manipulator serve meal to the user securely. As it can be observed from the figure 22, meal tray has no pattern information and therefore finding the stereo correspondence information and consequently 3D reconstruction becomes difficult. During the meal serving scenario, meal tray is available on a planar surface, i.e. either on the shelf of the refrigerator or inside a microwave or on the meal serving platform. Therefore the pose of meal tray on a planar surface only has 4 DoF (i.e. 3 DoF for the position of meal tray on the plane and 1 DoF for the rotation). Considering robot arm reaching positions, meal tray rotations are further restricted up to 180°. The rotation angle of meal tray is determined using two defined feature points of the red handle. The stereo feature points are extracted from the surface region of the red handle described in section 2.2.

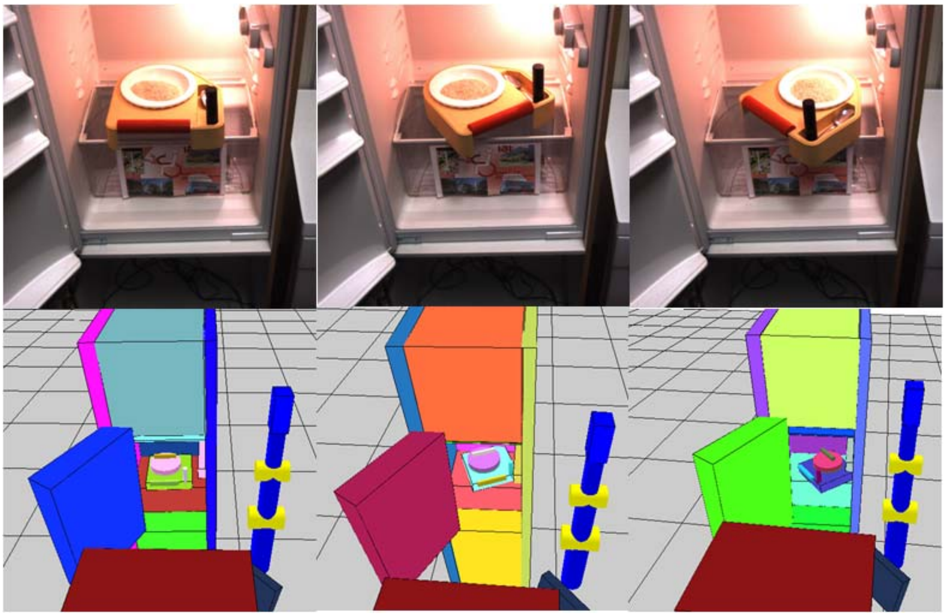


Fig. 23. Reconstructed meal tray poses from refrigerator

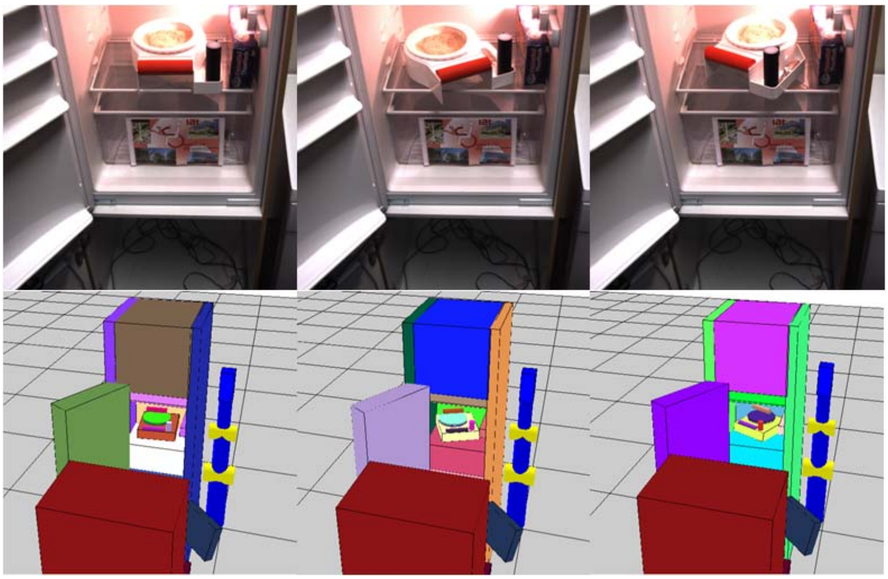


Fig. 24. Reconstructed meal tray poses from refrigerator

The binary image of the red handle represents a solid rectangle; the two end points are the mid points of short edges of minimum fitting rectangle. The rotation angle of the meal tray is determined according to the tracking method described in section 4.
Figures 22, 23 and 24 show the reconstructed meal tray images; top rows show the left stereo images and bottom images show reconstructed meal tray images. The meal tray shown in figure 24 is different from the one shown in figures 22 and 23. The meal serving scenario has been successfully executed numerous times at IAT and is presented in FRIEND III project meetings.

6. Conclusion

In autonomous object manipulation tasks, image processing and computer vision systems play vital role as the objects of interest shall be identified and localized autonomously. As it is always not possible to have calibration targets on the objects of interest that are needed to be localized and manipulated, special object recognition and localization methods which can be applied to such objects are required. The examples of such objects are domestic objects of our daily lives. The FRIEND system that is being developed at IAT supports the physically challenged people in their daily living scenarios such as serving a drink and serving a meal. The domestic objects used in FRIEND system belong to the category of texture less objects. In order to reconstruct the 3D models of these objects stereo vision based 3D object model reconstruction methods are investigated in this thesis work.
In order to reconstruct 3D object models selection of stereo feature points is required; as false stereo correspondences can result in large 3D reconstruction results, selection of stereo feature points for the texture less objects becomes a critical task. In case traditional stereo matching (i.e. intensity and feature based) techniques fail in finding stereo correspondences, geometry specific stereo feature points from high surface curvature regions is selected, stereo correspondences or unknown 3D coordinates information can be retrieved using constrained approaches which require priory object (or feature point in 3D) knowledge. Though strong limitations exist in the proposed tracking method, in contrast to the traditional methods object pose can be tracked using two object points; several applications could be benefited with such an approach. Serving meal and serving drink scenarios of FRIEND system have been successfully realised and currently being presented in international fairs using discussed stereo vision based object localization methods.

7. References

Birchfield.S & Tomasi.C. (1998). *Depth discontinuities by pixel-to-pixel stereo*, In: proceedings of 6th IEEE Int. Conf. Computer Vision, Mumbai, India, pp 1073-1080.

Corke. P. I. (1996). *Visual Control of Robots*. Research Studies Press Ltd, ISBN 0863802709, U.K

Garric. V & Devy. M, (1995). *Evaluation of the Calibration and Localization Methods for Visually Guided Grasping*, In: International Conference on Intelligent Robots and Systems, pp. 387-393. IEEE Press, Pittsburgh

Gilbert. S, Laganiere. R, Roth. G. (2006). Robust Object Pose Estimation from Feature Based Stereo, *IEEE Transactions on Measurement*, Vol 55, Number 4, NRC 48741

Hartley. R & Zisserman. A. (2000). Multiple View Geometry in Computer Vision, Cambridge Univ Press, ISBN 0521623049 ,UK

Horaud. R, Conio. B, Leboulleux. O & Lacolle. B. (1989). *An Analytic Solution for the Perspective 4-Point Problem*, In: Computer Vision and Pattern Recognition, pp 500—507. IEEE Press, San Diego

Hutchinson. S, Hager. G & Corke. P. (1996). A Tutorial on Visual Servo Control. *IEEE Transactions on Robotics and Automation*. Vol. 12, 651—670.

Klette.R , Schlüns.K & Koschan. A. (1998). *Computer Vision: Three-Dimensional Data from Images*. Springer, ISBN 9813083719, Singapore.

Lange. R. (2000). *3D Time-of-Flight Distance Measurement with Custom Solid-State Image Sensors in CMOS/CCD-Technology*, Ph.D. Thesis, ETH-Zürich.

Lange.R, Seitz. P, Biber. A, Schwarte. R. (1999). *Time-of-flight range imaging with a custom solid-state image sensor*, In: Proceedings of the SPIE, Vol. 3823, pp. 180-191, Munich.

Lin, M. (2002). *Surfaces with Occlusions from Layered Stereo, PhD thesi*s, Stanford University, California, United States.

Lowe. D. G. (1987) . Three-dimensional Object Recognition from Single Two-dimensional Images, *Artificial Intelligence*. Vol. 31, Number 3, pp: 355—395, ISSN:0004-3702

Rohn. J. (1992). Step size rule for unconstrained optimization, Vol 49, Number 4, Springer Wien, ISBN 0010-485X (Print) 1436-5057 (Online),

Shakunaga.T. (1991). `Pose estimation of jointed structures, In: Proceedings of Computer Vision and Pattern Recongnition, pp.566-572, Maui, USA.

Sterger. C, Ulrich. M & Wiedermann. C. (2007). *Machine Vision Algorithms and Applications*, WILEY-WCH, ISBN: 9783527407347, Germany.

Taylor. G. R & Kleeman. L. (2006) . *Visual Perception and Robotic Manipulation*, Springer-Verlag GmbH, ISBN: 9783540334545.

Tsai. R.Y. (1987): A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses, *IEEE Int. Journal Robotics and Automation*, Vol. 3, Number 4, pp. 323-344.

Yuan. J.S.C. (1989). A General Photogrammetric Method for Determining Object Position and Orientation, *IEEE Transactions on Robotics and Automation*, Vol 5, Number 2, pp: 129—142

Zhang, Z. (2000). A flexible New Technique for Camera Calibration, IEEE Transactions Pattern. Analysis and Machine Intelligence, Vol . 22, Number 11, pp: 1330-1334.

Yoon. Y, DeSouza. G. N, Kak A. C. (2003). *Real-time Tracking and Pose Estimation for Industrial Objects using Geometric Features*. In: Proceedings of the Int. Conference in Robotics and Automation, pp. 3473-3478. IEEE Press, Taiwan

Vision based Systems for Localization in Service Robots

Paulraj M.P. and Hema C.R.
Mechatronic Program,
School of Mechatronic Engineering,
Universiti Malaysia Perlis
Malaysia

1. Introduction

Localization is one of the fundamental problems of service robots. The knowledge about its position allows the robot to efficiently perform a service task in office, at a facility or at home. In the past, variety of approaches for mobile robot localization has been developed. These techniques mainly differ in ascertaining the robot's current position and according to the type of sensor that is used for localization. Compared to proximity sensors, used in a variety of successful robot systems, digital cameras have several desirable properties. They are low-cost sensors that provide a huge amount of information and they are passive so that vision-based navigation systems do not suffer from the interferences often observed when using active sound or light based proximity sensors. Moreover, if robots are deployed in populated environments, it makes sense to base the perceptional skills used for localization on vision like humans do.
In recent years there has been an increased interest in visual based systems for localization and it is accepted as being more robust and reliable than other sensor based localization systems. The computations involved in vision-based localization can be divided into the following four steps [Borenstein et al, 1996]:

(i) *Acquire sensory information*: For vision-based navigation, this means acquiring and digitizing camera images.
(ii) *Detect landmarks*: Usually this means extracting edges, smoothing, filtering, and segmenting regions on the basis of differences in gray levels, colour, depth, or motion.
(iii) *Establish matches between observation and expectation:* In this step, the system tries to identify the observed landmarks by searching in the database for possible matches according to some measurement criteria.
(iv) *Calculate position*: Once a match (or a set of matches) is obtained, the system needs to calculate its position as a function of the observed landmarks and their positions in the database.

2. Taxonomy of Vision Systems

There is a large difference between indoor and outdoor vision systems for robots. In this chapter we focus only on vision systems for indoor localization. Taxonomy of indoor based vision systems can be broadly grouped as [DeSouza and Kak, 2002]:

i. **Map-Based:** These are systems that depend on user-created geometric models or topological maps of the environment.
ii. **Map-Building-Based**: These are systems that use sensors to construct their own geometric or topological models of the environment and then use these models for localization.
iii. **Map-less:** These are systems that use no explicit representation at all about the space in which localization is to take place, but rather resort to recognizing objects found in the environment or to tracking those objects by generating motions based on visual observations.

In among the three groups, vision systems find greater potential in the map-less based localization. The map-less navigation technique and developed methodologies resemble human behaviors more than other approaches, and it is proposed to use a reliable vision system to detect landmarks in the target environment and employ a visual memory unit, in which the learning processes will be achieved using artificial intelligence. Humans are not capable of positioning themselves in an absolute way, yet are able to reach a goal position with remarkable accuracy by repeating a look *at the target and move* type of strategy. They are apt at actively extracting relevant features of the environment through a somewhat inaccurate vision process and relating these to necessary movement commands, using a mode of operation called visual servoing [DeSouza and Kak, 2002].

Map-less navigation include systems in which navigation and localization is realized without any prior description of the environment. The localization parameters are estimated by observing and extracting relevant information about the elements in the environment. These elements can be walls, objects such as desks, doorways, etc. It is not necessary that absolute (or even relative) positions of these elements of the environment be known. However, navigation and localization can only be carried out with respect to these elements.

Vision based localization techniques can be further grouped based on the type of vision used namely, passive stereo vision, active stereo vision and monocular vision. Examples of these three techniques are discussed in detail in this chapter.

3. Passive Stereo Vision for Robot Localization

Making a robot see obstacles in its environment is one of the most important tasks in robot localization and navigation. A vision system to recognize and localize obstacles in its navigational path is considered in this section. To enable a robot to see involves at least two mechanisms: sensor detection to obtain data points of the obstacle, and shape representation of the obstacle for recognition and localization. A vision sensor is chosen for shape detection of obstacle because of its harmlessness and lower cost compared to other sensors such as laser range scanners. Localization can be achieved by computing the distance of the object from the robot's point of view. Passive stereo vision is an attractive technique for distance measurement. Although it requires some structuring of the environment, this method is appealing because the tooling is simple and inexpensive, and in many cases already existing cameras can be used. An approach using passive stereo vision to localize objects in a controlled environment is presented.

3.1 Design of the Passive Stereo System

The passive stereo system is designed using two digital cameras which are placed on the same *y*-plane and separated by a base length of 7 cm in the *x*-plane. Ideal base lengths vary from 7 cm to 10 cm depicting the human stereo system. The height of the stereo sensors depends on the size of objects to be recognized in the environment, in the proposed design the stereo cameras are placed at a height of 20 cm. Fig. 1 shows the design of mobile robot with passive stereo sensors. It is important to note both cameras should have the same view of the object image frame to apply the stereo concepts. An important criterion of this design is to keep the blind zone to a minimal for effective recognition as shown in Fig.2.

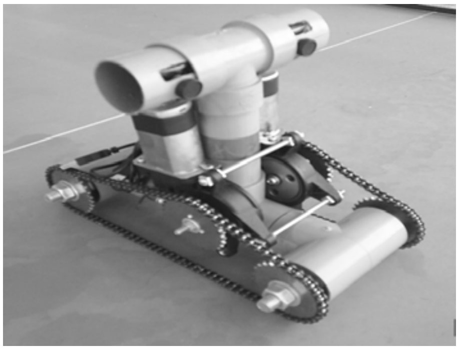


Fig. 1. A mobile robot design using passive stereo sensors

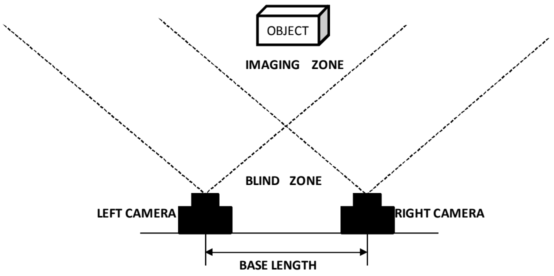


Fig. 2 Experimental setup for passive stereo vision

3.2 Stereo Image Preprocessing

Color images acquired from the left and the right cameras are preprocessed to extract the object image from the background image. Preprocessing involves resizing, grayscale conversion and filtering to remove noise, these techniques are used to enhance, improve or otherwise alter an image to prepare it for further analysis. The intension is to remove noise, trivial information or information that will not be useful for object recognition. Generally object images are corrupted by indoor lighting and reflections. Noise can be produced due to low lighting also. Image resizing is used to reduce the computational time, a size of 320 by 240 is chosen for the stereo images. Resized images are converted to gray level images to reduce the pixel intensities to a gray scale between 0 to 255; this further reduces the computations required for segmentation.

Acquired stereo images do not have the same intensity levels; there is considerable difference in the gray values of the objects in both left and right images due to the displacement between the two cameras. Hence it is essential to smooth out the intensity of both images to similar levels. One approach is to use a regional filter with a mask. This filter filters the data in the image with the 2-D linear Gaussian filter and a mask. The mask image is the same size as the original image. Hence for the left stereo image, the right stereo image can be chosen as the mask and vice versa. This filter returns an image that consists of filtered values for pixels in locations where the mask contains 1's, and unfiltered values for pixels in locations where the mask contains 0's. The intensity around the obstacle in the stereo images is smoothened by the above process.

A median filter is applied to remove the noise pixels; each output pixel contains the median value in the M-by-N neighborhood [M and N being the row and column pixels] around the corresponding pixel in the input image. The filter pads the image with zeros on the edges, so that the median values for the points within [M N]/2 of the edges may appear distorted [Rafael, 2002]. The M -by- N is chosen according to the dimensions of the obstacle. A 4 x 4 matrix was chosen to filter the stereo images. The pre-processed obstacle images are further subjected to segmentation techniques to extract the obstacle image from the background.

3.3 Segmentation

Segmentation involves identifying an obstacle in front of the robot and it involves the separation of the obstacle from the background. Segmentation algorithm can be formulated using the grey value obtained from the histogram of the stereo images. Finding the optimal threshold value is essential for efficient segmentation. For real-time applications, automatic determination of threshold value is an essential criterion. To determine this threshold value a weighted histogram based algorithm is proposed which uses the grey levels of the image from the histogram of both the stereo images to compute the threshold. The weighted histogram based segmentation algorithm is detailed as follows [Hema et al, 2006]:

Step 1: The histogram is computed from the left and right gray scale images for the gray scale values of 0 to 255.

Counts $a(i)$, $i=1,2,3,\ldots,256$
where $a(i)$ represents the number of pixels with gray scale value of (i-1) for the left image.

Counts $b(i)$, $i=1,2,3,\ldots,256$
where $b(i)$ represents the number of pixels with gray scale value (i-1) for the right image.

.

Step 2: Compute the logarithmic weighted gray scale value of the left and right image as

$$ta\ (i) = \log(\ count\ a\ (i)) * (i\text{-}1) \tag{1}$$

$$tb\ (i) = \log(\ count\ b\ (i)) * (i\text{-}1) \tag{2}$$

where $i = 1,2,3,\ldots,256$

Step 3: Compute the logarithmic weighted gray scale

$$tam = \frac{1}{256}\sum_{i=1}^{256} ta(i) \tag{3}$$

$$tbm = \frac{1}{256}\sum_{i=1}^{256} tb(i) \tag{4}$$

Step 4: Compute T = min(*tam, tbm*). The minimum value of '*tam*' and '*tbm*' is assigned as the threshold.

Threshold of both the stereo images are computed separately and the min value of the two thresholds is applied as the threshold to both the images. Using the computed threshold value, the image is segmented and converted into a binary image.

3.4 Obstacle Localization

Localization of the obstacle can be achieved using the information from the stereo images. One of the images for example, the left image is considered as the reference image. Using the reference image, the x and y co-ordinates is computed by finding the centroid of the obstacle image. The z co-ordinate can be computed using the unified stereo imaging principle proposed in [Hema et al, 2007]. The unified stereo imaging principle uses a morphological 'add' operation to add the left and right images acquired at a given distance. The size and features of the obstacle in the added image varies in accordance with the distance information. Hence, the features of the added image provide sufficient information to compute the distance of the obstacle. These features can be used to train a neural network to compute the distance (z). Fig.3 shows images samples of the added images and the distance of the obstacle images with respect to the stereo sensors. The features extracted from the added images are found to be good candidates for distance computations using neural networks [Hema et al, 2007].

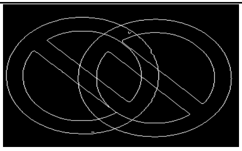
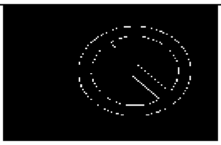
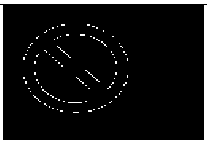
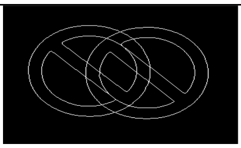
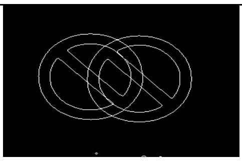
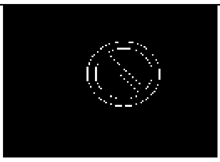
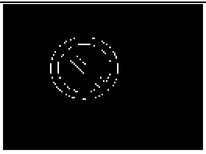
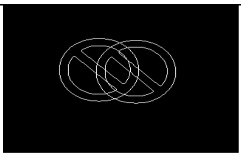
Left Image	Right Image	Add Image	Distance (cm)
			45
			55
			65
			85

Fig. 3. Sample Images of added stop symbol images and the distance of the obstacle image from the stereo sensors.

The x, y and z co-ordinate information determined from the stereo images can be effectively used to locate obstacles and signs which can aid in collision free navigation in an indoor environment.

4. Active Stereo Vision for Robot Orientation

Autonomous mobile robots must be designed to move freely in any complex environment. Due to the complexity and imperfections in the moving mechanisms, precise orientation and control of the robots are intricate. This requires the representation of the environment, the knowledge of how to navigate in the environment and suitable methods for determining the orientation of the robot. Determining the orientation of mobile robots is essential for robot path planning; overhead vision systems can be used to compute the orientation of a robot in a given environment. Precise orientation can be easily estimated, using active stereo vision concepts and neural networks [Paulraj et al, 2009]. One such active stereo vision system for determining the robot orientation features from the active stereo vision system in indoor environments is described in this section.

4.1 Image Acquisition

In active stereo vision two are more cameras are used, wherein the cameras can be positioned to focus on the same imaging area from different angles. Determination of the position and orientation of a mobile robot, using vision sensors, can be explained using a simple experimental setup as shown in Fig.4. Two digital cameras using the active stereo concept are employed. The first camera (C1) is fixed at a height of 2.1 m above the floor level in the centre of the robot working environment. This camera covers a floor area of size 1.7m length (L) and 1.3m width (W). The second camera (C2) is fixed at the height (H2) of 2.3 m above the ground level and 1.2 m from the Camera 1. The second camera is tilted at an angle (θ_2) of 22.5^0.

The mobile robot is kept at different positions and orientation and the corresponding images (O_{a1} and O_{b1}) are acquired using the two cameras. The experiment is repeated for 180 different orientation and locations. For each mobile robot position, the angle of orientation is also measured manually. The images obtained during the i^{th} orientation and position of the robot is denoted as (O_{ai}, O_{bi}). Sample of images obtained from the two cameras for different position and orientation of the mobile robot are shown in Fig.5.

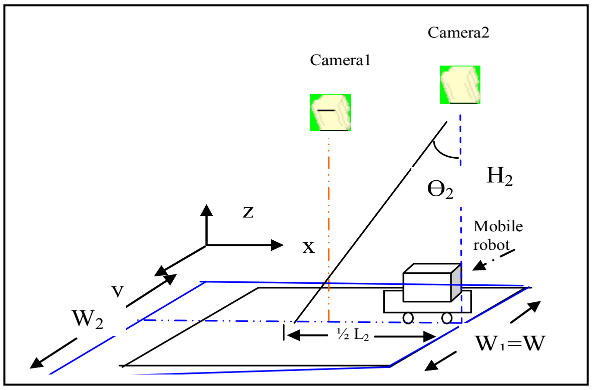
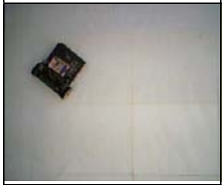
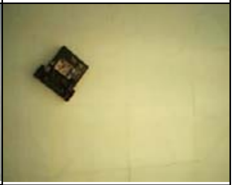


Fig. 4. Experimental Setup for the Active Stereo Vision System

Camera 1	Camera 2
	

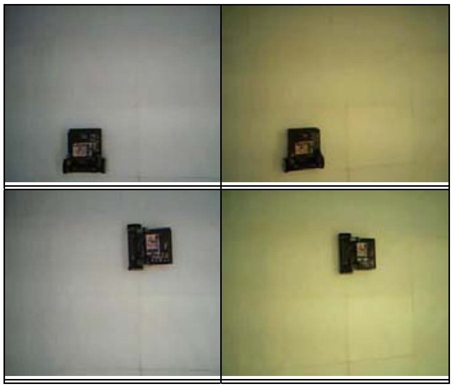


Fig. 5. Samples of images captured at different orientations using two cameras

4.2 Feature Extraction

As the image resolution causes considerable delay while processing, the images are resized to 32 x 48 pixels and then converted into gray-scale images. The gray scale images are then converted into binary images. A simple image composition is made by multiplying the first image with the transpose of the second image and the resulting image I_u is obtained. Fig.6 shows the sequence of steps involved for obtaining the composite image I_u. The original images and the composite image are fitted into a rectangular mask and their respective local images are obtained. For each binary image, sum of pixel value along the rows and the columns are all computed. From the computed pixel values, the local region of interest is defined. Fig. 7 shows the method of extracting the local image. Features such as the global centroid, local centroid, and moments are extracted from the images and used as a feature to obtain their position and orientation. The following algorithm illustrates the method of extracting the features from the three images.

Feature Extraction Algorithm:

1) Resize the original images O_a, O_b.
2) Convert the resized images into gray-scale images and then to binary images. The resized binary images are represented as I_a and I_b.
3) Fit the original image I_a into a rectangular mask and obtain the four coordinates to localize the mobile robot. The four points of the rectangular mask are labeled and cropped. The cropped image is considered as a local image (I_{al}) .
4) For the original image I_a determine the global centroid (G_{ax}, G_{ay}), area (G_{aa}), perimeter (G_{ap}). Also for the localized image I_{al}, determine the centroid (L_{ax}, L_{ay}) row sum pixel values (L_{ar}) , column sum pixel values (L_{ac}), row pixel moments (L_{arm}) column pixel moments (L_{acm}).
5) Repeat step 3 and 4 for the original image I_b and determine the parameters G_{bx}, G_{by}, G_{ba}, G_{bp}, L_{bx}, L_{by}, L_{br}, L_{bc}, L_{brm} and L_{bcm}.
6) Perform stereo composition: $I_u = I_a \times I_b^T$. (where T represents the transpose operator).
7) Fit the unified image into a rectangular mask and obtain the four coordinates to localize the mobile robot. The four points of the rectangular mask are labeled and cropped and labeled and cropped. The cropped image is considered as a local image.

8) From the composite global image, the global centroid (G_{ux}, G_{uy}), area (G_{ua}), perimeter (G_{up}) are computed.
9) From the composite local image, the local centroid (L_{ux}, L_{uy}) row sum pixel values (L_{ur}) , column sum pixel values (L_{uc}), row pixel moments (L_{urm}) column pixel moments (L_{ucm}) are computed.

The above features are associated to the orientation of the mobile robot.

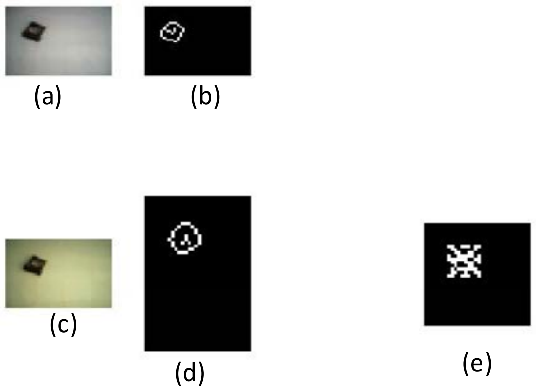


Fig. 6. (a) Resized image from first camera with 32 x 48 pixel, (b) Edge image from first camera, (c) Resized image from second camera with 48 x 32 pixel, (d) Edge image from second camera with transposed matrix (e)Multiplied images from first and second cameras with 32 x 32 pixel.

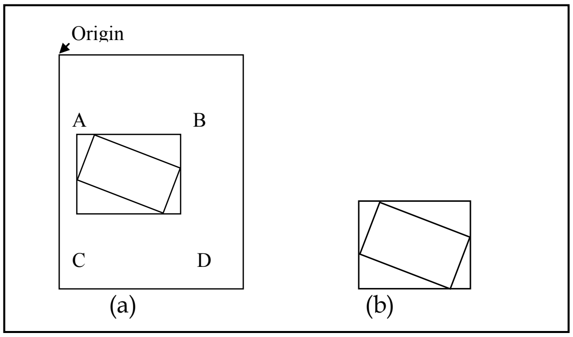


Fig. 7. Extraction of local image (a) Global image (b) Local or Crop image.

5. Hybrid Sensors for Object and Obstacle Localization in Housekeeping Robots

Service robots can be specially designed to help aged people and invalids to perform certain housekeeping tasks. This is more essential to our society where aged people live alone. Indoor service robots are being highlighted because of their potential in scientific, economic and social expectations [Chung et al, 2006; Do et al, 2007]. This is evident from the growth of service robots for specific service tasks around home and work places. The capabilities of the mobile service robot require more sensors for navigation and task performance in an unknown environment which requires sensor systems to analyze and recognize obstacles and objects to facilitate easy navigation around obstacles. Causes of the uncertainties include people moving around, objects brought to different positions, and changing conditions.
A home based robot, thus, needs high flexibility and intelligence. A vision sensor is particularly important in such working conditions because it provides rich information on surrounding space and people interacting with the robot. Conventional video cameras, however, have limited fields of view. Thus, a mobile robot with a conventional camera must look around continuously to see its whole surroundings [You, 2003]. This section highlights a monocular vision based design for a housekeeping robot prototype named ROOMBOT, which is designed using a hybrid sensor system to perform housekeeping tasks, which includes recognition and localization of objects. The functions of the hybrid vision system alone are highlighted in this section.
The hybrid sensor system combines the performance of two sensors namely a monocular vision sensor and an ultrasonic sensor. The vision sensor is used to recognize objects and obstacles in front of the robot. The ultrasonic sensor helps to avoid obstacles around the robot and to estimate the distance of a detected object. The output of the sensor system aids the mobile robot with a gripper system to pick and place the objects that are lying on the floor such as plastic bags, crushed trash paper and wrappers.

5.1 ROOMBOT Design

The ROOMBOT consists of a mobile platform which has an external four wheeled drive found to be suitable for housekeeping robots; the drive system uses two drive wheels and two castor wheels, which implement the differential drive principle. The left and right wheels at the rear side of the robot are controlled independently [Graf et al, 2001]. The robot turning angle is determined by the difference of linear velocity between the two drive wheels. The robot frame has the following dimensions 25cm (width) by 25cm (height) and 50cm (length). The robot frame is layered to accommodate the processor board and control boards. The hybrid sensor system is placed externally to optimize the area covered. The housekeeping robot is programmed to run along a planned path. The robot travels at an average speed of 0.15m/s. The navigation system of the robot is being tested in an indoor environment. The robot stops when there is an object in front of it at the distance of 25cm. It is able to perform 90°-turns when an obstacle is blocking its path. The prototype model of the robot is shown in Fig.8.

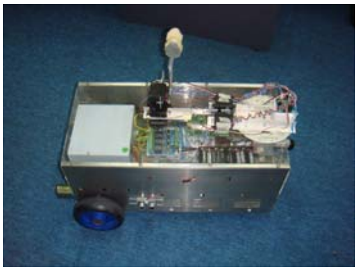


Fig. 8. Prototype model of the housekeeping robot.

5.2 Hybrid Sensor System

The hybrid sensor system uses vision and ultrasonic sensors to facilitate navigation by recognizing obstacles and objects on the robot's path. One digital camera is located on the front panel of the robot at a height of 17 cm from the ground level. Two ultrasonic sensors are also placed below the camera as shown in Fig.9 (a). The ultrasonic sensors below the camera is tilted at an angle of 10 degrees to facilitate the *z* co-ordinate computations of the objects as shown in Fig.9(b). Two ultrasonic sensors are placed on the sides of the robot for obstacle detection (Fig.9(c)). The two ultrasonic sensors in the front are used for detecting objects of various sizes and to estimate the *y* and *z* co-ordinates of objects.

The ultrasonic system detects obstacles / objects and provides distance information to the gripper system. The maximum range of detection of the ultrasonic sensor is 3 m and the minimum detection range is 3 cm. Due to uneven propagation of the transmitted wave, the sensor is unable to detect in certain conditions [Shoval & Borenstein 2001]. In this study, irregular circular objects are chosen for height estimation. Therefore the reflected wave is not reflected from the top of the surface. This will contribute to small error which is taken into account by the gripper system.

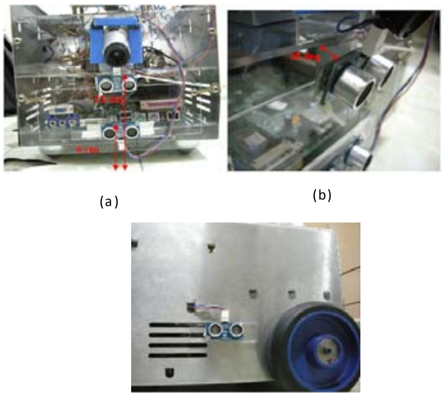

(a)
(b)
(c)

Fig. 8. Vision and ultrasonic sensor locations (a) vision and two ultrasonic sensors in the front panel of the robot, (b) ultrasonic sensor with 10 degree tilt in the front panel, (c) ultrasonic sensor located on the sides of the robot.

5.3 Object Recognition

Images of objects such as crushed paper and plastic bags are acquired using the digital camera. Walls, furniture and cardboard boxes are used for the obstacle images. An image database is created with objects and obstacles in different orientation and acquired at different distances. The images are dimensionally resized to 150 x 150 sizes to minimize memory and processing time. The resized images are processed to segment the object and suppress the background. Fig.9 shows the image processing technique employed for segmenting the object. A simple feature extraction algorithm is applied to extract the relevant features which can be fed to a classifier to recognize the objects and obstacles. The feature extraction algorithm uses the following procedure:

Step1. Acquired image is resized to 150 x 150 pixel sizes to minimize memory and processing time.
Step2. Resized images are converted to binary images using the algorithm detailed in section 3.3. This segments the object image from the background
Step3. Edge images are extracted from the binary images to further reduce the computational time.
Step4. The singular values are extracted from the edge images using singular value decomposition on the image matrix.

The singular values are used to train a simple feed forward neural network to recognize the objects and the obstacle images [Hong, 1991; Hema et al, 2006]. The trained network is used for real-time recognition during navigation. Details of the experiments can be found [Hema et al, 2009].

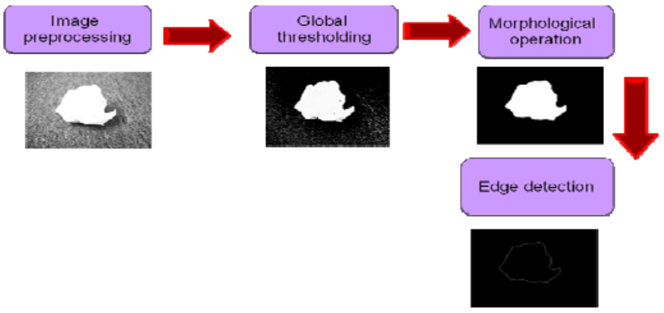


Fig. 9. Flow diagram for Image segmentation

5.4 Object Localization

Object localization is essential for pick and place operation to be performed by the gripper system of the robot. In the housekeeping robot, the hybrid sensor system is used to localize the objects. Objects are recognized by the object recognition module; using the segmented object image the *x* co-ordinate of the object is computed. The distance derived from the two ultrasonic sensors in the front panel is used to compute the *z* co-ordinate of the object as shown in Fig. 10. The distance measurement of the lowest ultrasonic sensors gives the *y* co-ordinate of the object. The object co-ordinate information is passed to the gripper system to perform the pick and place operation. Accuracy of 98% was achievable in computing the *z* co-ordinate using the hybrid vision system.

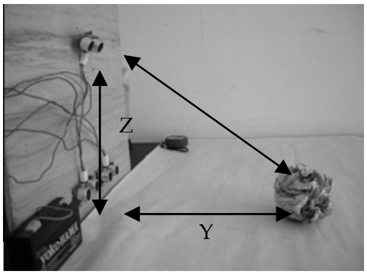


Fig. 10. Experimental setup to measure the *z* co-ordinate.

The ROOMBOT has an overall performance of 99% for object recognition and localization. The hybrid sensor system proposed in this study can detect and locate objects like crushed paper, plastic and wrappers. Sample images of the experiment for object recognition and pick up are shown in Fig.11.

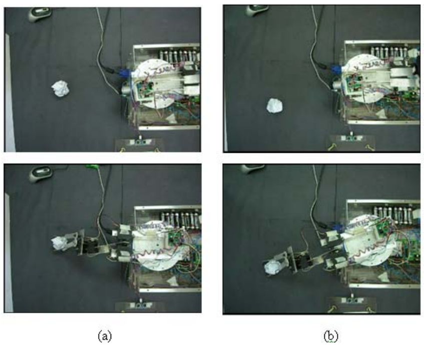


Fig. 11. picking of trash paper based on computation of the object co-ordinates (a) location 1 (b) location 2.

6. Conclusion

This chapter proposed three vision based methods to localize objects and to determine the orientation of robots. Active and passive stereo visions along with neural networks prove to be efficient techniques in localization of objects and robots. The unified stereo vision system discussed in this chapter depicts the human biological stereo vision system to recognise and localize objects. However a good database of typical situations is necessary is implement the system. Combining vision sensors with ultrasonic sensors is an effective method to combine the information from both sensors to localize objects. All the three vision systems were tested in real-time situations and their performance were found to be satisfactory. However the methods proposed are only suitable for controlled indoor environments, further investigation is required to extend the techniques to outdoor and challenging environments.

7. References

Borenstein J. Everett H.R. and Feng L. (1996) Navigating Mobile Robots: Systems and Techniques, eds. Wellesley, Mass.: AK Peters,.

Chung W, Kim C. and Kim M.(2006) Development of the multi-functional indoor service robot PSR systems" Autonomous Robot Journal, pp.1-17,

DeSouza G.N and Kak A.C.(2002) Vision for Mobile Robot Navigation: A Survey, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24, No. 2, February 2

Do Y. Kim G. and Kim J (2007) Omni directional Vision System Developed for a Home Service Robot" 14th International Conference on Mechatronic and Machine Vision in Practice.

Graf B. Schraft R. D. and Neugebauer J. (2001) A Mobile Robot Platform for Assistance and Entertainment" Industrial Robot: An International Journal, pp.29-35.

Hema C.R., Paulraj M.P., Nagarajan R. and Yaacob S.(2006) "Object Localization using Stereo Sensors for Adept SCARA Robot" Proc. of IEEE Intl. Conf. on Robotics, Automation and Mechatronics, , pp.1-5,

Hema C.R. Paulraj M.P. Nagarajan R. and Yaacob S. (2007) Segmentation and Location Computation of Bin Objects. International Journal of Advanced Robotic Systems Vol. 4 No.1, pp.57-62.

Hema C.R. Lam C.K. Sim K.F. Poo T.S. and Vivian S.L. (2009) Design of ROOMBOT- A hybrid sensor based housekeeping robot", International Conference On "Control, Automation, Communication And Energy Conservation, India , June 2 , pp.396-400.

Hong Z. (1991) Algebraic Feature Extraction of Image for Recognition", IEEE Transactions on Pattern Recognition, vol. 24 No. 3 pp: 211-219.

Paulraj M.P. Fadzilah H. Badlishah A.R. and Hema C. R. (2009) Estimation of Mobile Robot Orientation Using Neural Networks. International Colloquium on Signal Processing and its Applications, Kuala Lumpur, 6-8 March, pp. 43-47.

Shoval S. and Borenstein D. (2001) Using coded signal to benefit from ultrasonic sensor crosstalk in mobile robot obstacle avoidance. IEEE International Conference on Robotics and Automation, Seoul, Korea, May 21-26, pp. 2879-2884.

You J. (2003) Development of a home service robot 'ISSAC'," Proc. IEEE/RSJ IROS, pp. 2630-2635.

Floor texture visual servo using multiple cameras for mobile robot localization

Takeshi Matsumoto, David Powers and Nasser Asgari
Flinders University
Australia

1. Introduction

The study of mobile robot localization techniques has been of increasing interest to many researchers and hobbyists as accessibility to mobile robot platforms and sensors have improved dramatically. The field is often divided into two categories, local and global localization, where the former is concerned with the pose of the robot with respect to the immediate surroundings, while the latter deals with the relationship to the complete environment the robot considers. Although the ideal capability for localization algorithms is the derivation of the global pose, the majority of global localization approaches make use of local localization information as the foundation.
The use of simple kinematic models or internal sensors, such as rotational encoders, often have limitations in accuracy and adaptability in different environments due to the lack of feed back information to correct any discrepancies between the motion model and the actual motion. Closed loop approaches, on the other hand, allow for more robust pose calculations using various sensors to observe the changes in the environment as the robot moves around. One of these sensors of increasing interest is the camera, which has become more affordable and precise in being able to capture the structure of the scene.
The proposed techniques include the investigation of the issues in using multiple off-the-shelf webcams mounted on a mobile robot platform to achieve a high precision local localization in an indoor environment (Jensfelt, 2001). This is achieved through synchronizing the floor texture tracker from two cameras mounted on the robot. The approach comprises of three distinct phases; configuration, feature tracking, and the multi-camera fusion in the context of pose maintenance.
The configuration phase involves the analysis of the capabilities of both the hardware and software components that are integrated together while considering the environments in which the robot will be deployed. Since the coupling between the algorithm and the domain knowledge limits the adaptability of the technique in other domains, only the commonly observed characteristics of the environment are used. The second phase deals with the analyses of the streaming images to identify and track key features for visual servo (Marchand & Chaumette, 2005). Although this area is a well studied in the field of image processing, the performance of the algorithms are heavily influenced by the environment. The last phase involves the techniques for the synchronizing multiple trackers and cameras.

2. Background

2.1 Related work

The field of mobile robot localization is currently dominated by global localization algorithm (Davison, 1998; Se et al., 2002; Sim & Dudek, 1998; Thrun et al., 2001; Wolf et al., 2002), due to the global pose being the desired goal. However, a robust and accurate local localization algorithm has many benefits, such as faster processing time, less reliability on the landmarks, and they often form the basis for global localization algorithms.
Combining the localization task with image processing allows the use of many existing algorithms for extracting information about the scene (Ritter, & Wilson, 1996; Shi & Tomasi, 1994), as well as being able to provide the robot with a cheap and precise sensor (Krootjohn, 2007). Visual servo techniques have often been implemented on stationary robotics to use the visual cues for controlling its motion. The proposed approach operates in a similar way, but observes the movement of the ground to determine the pose of itself.
The strategy is quite similar to how an optical mouse operates (Ng, 2003), in that the local displacement is accumulated to determine the current pose of the mouse. However, it differs on several important aspects like the ability to determine rotation, having less tolerance for errors, as well as being able to operate on rough surfaces.

2.2 Hardware

The mobile robot being used is a custom built model to be used as a platform for incrementally integrating various modules to improve its capabilities. Many of the modules are developed as part of undergraduate students' projects, which focus on specific hardware or software development (Vilmanis, 2005). The core body of the robot is a cylindrical differential drive system, designed for indoor use. The top portion of the base allows extension modules to be attached in layers to house different sensors while maintaining the same footprint, as shown in Fig. 1.

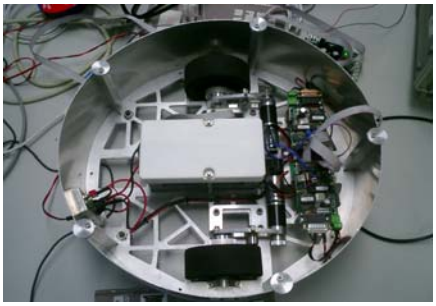


Fig. 1. The robot base, the rotational axis of the wheels align with the center of the robot

The boards mounted on the robot control the motors, the range finding sensors, as well as relaying of commands and data through a serial connection. To allow the easy integration of off-the-shelf sensors, a laptop computer is placed on the mobile robot to handle the coordination of the modules and to act as a hub for the sensors.

2.3 Constraints

By understanding the systems involved, the domain knowledge can be integrated into the localization algorithm to improve its performance. Given that the robot only operates in indoor environments, assumptions can be made about the consistency of the floor. On a flat surface, the distance between the floor and a camera on the robot remains constant. This means the translation of camera frame motion to robot motion can be easily calculated.

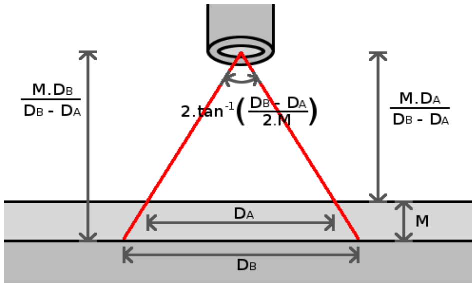
Another way to simplify the process is to restrict the type of motion that is observed. When rotation of the robot occurs, the captured frame become difficult to compare due to the blending that occurs between the pixels as they are captured on a finite array of photo sensors. To prevent this from affecting the tracking, an assumption can be made based on the frame rate, the typical motions of the mobile robot, life time of the features, and the position of the camera. By assuming that the above ideas amounts to minimised rotation, it is possible to constrain the feature tracking to only detect translations.

3. Camera configuration

3.1 Settings

The proposed approach assumes that the camera is placed at a constant elevation off the ground, thus reducing the image analysis to a simple 2D problem. By observing the floor from a wider perspective, the floor can be said to be flat, as the small bumps and troughs become indistinguishable.

Measuring the viewing angle of the camera can be achieved as per fig. 2, which can be used to derive the width and height of the captured frame at the desired elevation. This information can be used to determine the elevation of the camera where the common bumps, such as carpet textures, become indistinguishable. A welcome side-effect of increasing the elevation of the camera is the fact that it can avoid damages to the camera from obstacles that could scrape the lens.



$\frac{M.D_B}{D_B - D_A}$
$2.\tan^{-1}\left(\frac{D_B - D_A}{2.M}\right)$
$\frac{M.D_A}{D_B - D_A}$
D_A
M
D_B

Fig. 2. Deriving the viewing angle, the red line represents the bounds of the view

Since the precision of the frame tracking is relative to the elevation of the camera, raising the height of the camera reduces the accuracy of the approach. There is also an additional issue to consider with regards to being able to observe the region of interest in consecutive frames, which also relates to the capture rate and the speed of the robot.

Resolving the capture rate issue is the simplest, as this also relates to the second constraint, which states that no rotation can occur between the frames. By setting the frame rate to be as fast as possible, the change between the frames is reduced. Most webcams have a frame rate

of around 30Hz, which is fast enough for their normal usage. In comparison, an optical mouse may have a frame rate in the order of several thousand frames per second. This allows the viewing area to remain small, which leads to the optical mouse being able to capture a very small and detailed view of the ground. Although this difference places the webcam at some disadvantage, the increased viewing area allows the option of increasing the speed of the robot or reducing the search area for the tracking.
Since the tracking of the region of interest must be conducted reliably and efficiently, the speed of the robot must be balanced with the search area. Although this relationship can be improved with an implementation of a prediction algorithm to prune some of the search area, it is important to note the worst case scenarios instead of the typical cases as it can potentially hinder the other modules.
Since the motion blur causes the appearance of previously captured region of interest to change, it is desirable to reduce the exposure time. In environments with dynamically changing lighting conditions, this is not always plausible. This issue is enhanced by the fact that indoor ambient lighting conditions do not permit for much light to be captured by the camera, especially one that is pointing closely to the ground. A simple strategy to get around this issue is to provide a permanent light source near by. Out of several colours and lighting configurations, as shown in fig. 3, the uniformly distributed white light source was selected. The white light source provides a non-biased view of the floor surface, while the uniformly distributed light allows the same region to appear in a similar way from any orientation. To obtain the uniformly distributed light, the light was scattered using a crumpled aluminium foil and bouncing the light against it.

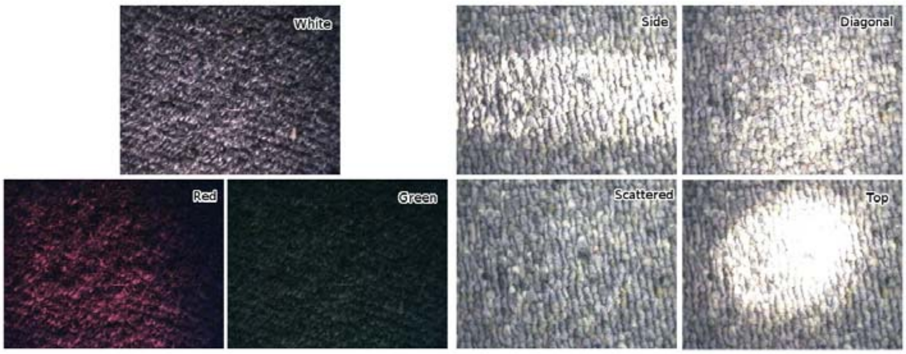


Fig. 3. Colour and shape of light source, left image shows various colours, while the right shows the shape of light

The majority of this flickering from the indoor lights can be eliminated by adjusting the exposure time. However, there is a related issue with sunlight if there are windows present, which can result in significantly different appearances when in and out of shadows. A simple way to solve this is to block the ambient light by enclosing the camera and light source configuration. This can be a simple cardboard box with a slight gap to the ground, enough to allow rough floor textures to glide underneath without hitting it.
One other important settings is with regards to the physical placement of the camera. Since the approach requires the view of the ground, the placement of the camera is limited to the outer borders of the robot, inside the core body, or suspended higher or further out using a

long arm. The use of an arm must consider the side effects like shaking and possible collision with obstacles, while placing the camera inside the main body restricts the distance to the ground. This can severely reduce the operation speed of the robot and also make the placement of the light source difficult. Another issue to consider is the minimum focal distance required on some webcams, which can sometimes start at around 9 cm.

This leaves the option of placing the camera on the outer perimeter of the robot, which allows enough stability, as well as some protection from collisions using the range finders. The camera has been currently placed at the minimum focal distance of the camera, which is at 9 cm from the ground.

3.2 Noise characteristics

With the camera configured, filters can be put in place to remove artefacts and enhance the image. There are many number of standard clean-up filters that can be applied, but without knowing the type of noise the image contains, the result may not be as effective or efficient in terms of processing cost. There are many different sources of unwanted artefacts, including the lens quality, the arrangement and quality of the photo sensors, and the scatter pattern of the light from the ground. Many of these errors are dependant on the camera, thus require individual analysis to be carried out.

One of the most common artefacts found on webcams is the distortion of the image in the outer perimeters, especially on older or cheaper cameras. In some of the newer models, there is a distortion correction mechanism built into the camera driver, which reduces the warping effect quite effectively. Without this, transformations are required in software to correct the warping by first measuring the amount of the warp. Although this calibration process is only required once, the correction is constantly required for each frame, costing valuable processing time.

By knowing the characteristics of the robot, such as the speed and the area viewed by each frame, it is possible to identify regions that will not be used. This information allows portions of the captured image to be discarded immediately, thus avoiding any processing required to de-warp the region. The effects of the lens distortion is stronger in the outer regions of the frame, thus by only using the center of the image, the outer regions can simply be discarded. The regions towards the center may also be slightly warped, but the effect is usually so small that corrections involve insignificant interpolation with the neighbours that it can simply be ignored.

When observing the streams of images, it was noted that the speckles of noise were more noticeable in certain regions of the image, as well as when certain colours were being displayed. To investigate this further, the characteristics of the photo-sensors on the cameras were determined when observing various colours. The first of these tests involved the identification of the noise prone or defective regions.

Since the gain and contrast settings on the camera can shift or trim the intensity value, the two extreme ends of the intensity scale were chosen. By saturating the camera with a bright light source, it was quite easy to capture a uniform white image. However, capturing a completely black image proved to be slightly more difficult due to the camera not updating when no light was observed. To get around this issue, slightly over half of the view was blocked to capture the behaviour for one side, then repeated for the other side before they were merged. The experiment showed that there was a significant fluctuation in the intensity when observing a dark image compared to a bright image, which did not seemed to be affected by the random fluctuation at all when saturated.

The noise characteristics of the black image can be seen in fig. 4, which shows distinctive patterns and positions of where the noise occurs. The duration of the sampling was set to 1,000 frames.

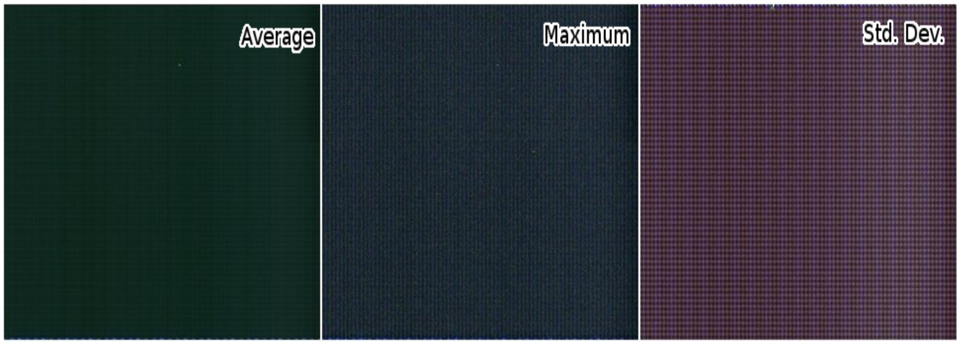


Fig. 4. Noise characteristics, left shows the average, middle shows the maximum, and the right shows the standard deviation

To explore the idea of intensity based noise, a second experiment was carried out by displaying a gradient of white to black and observing the amount of fluctuation detected at the various intensities, as shown in fig. 5. The base intensity was determined by the average intensity of the pixel, which was taken over 1,000 frames. The trend shows the relationship between the base intensity and the amount of fluctuation that is encountered. The waves that can be observed is due to the bands that can be seen in the captured image, as well as artefacts from the codec, which will be discussed later. The relationship allows a general model to be constructed for determining the expected noise at a particular intensity. An interesting observation that was made was that the noise caused small fluctuations from the intended intensity, thus allowing the possibility of thresholds to determine if the change in the intensity is due to noise or change in the scene.

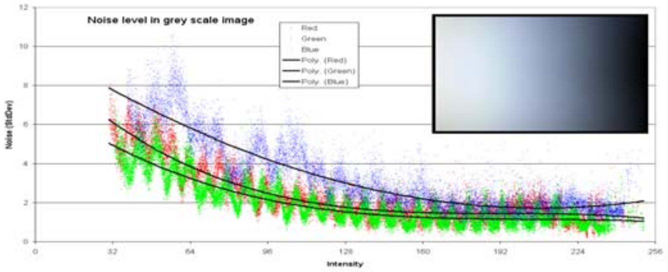


Fig. 5. Noise level against intensity, the grey scale image in the top right is the image captured by the camera

The last of the noise to be considered is the radial bands that can form due to the arrangement of the light source and the dispersion of the light on reflective surfaces, as shown in fig. 6. The left image shows the original snapshot of a table top, the middle image shows the stretched version across the full intensity range, while the right image shows the standard deviation, which has also been stretched. The bands that can be seen is the camera's tendency to approximate to the surrounding colour, which occurs as part of the codex.

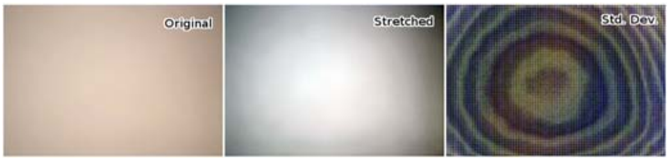


Fig. 6. Radial intensity shift, this noise often occurs on glossy surfaces

3.3 Image filter

The first type of filters to be investigated is the range modifiers, which controls the minimum and the maximum intensities, which indirectly shifts the intensities depending on the position or the captured intensity. Since the maximum intensity could be reached, there is no need to modify this intensity value. However, the minimum value was often not zero for many of the pixels, thus required some further thought.

The minimum value may be related to the issue with not being able to capture a pitch black image or residuals and noise generated within the hardware to not allow the distinction between pitch black and almost pitch black objects. Since the minimum value is quite small, often being less than 1/64th of the full range, the significance of this offset is not an important issue to warrant the costly operations required to process it.

With regards to the radial shift in the intensity, it is possible to determine if a correction is required by observing a steady increase in the intensity value towards the center of the image. However, it is better to prevent the issue from occurring than correcting it, thus the light source can be better configured to scatter the light more evenly.

A commonly used filter to clean up the sensor noise is a spatial or temporal filter to blend the neighbouring pixel's intensities. This reduces the random noise due to the object's size and position consistency across multiple frames. One of the major drawbacks of this approach is its behaviour to blur the whole the image since the blending is applied in inappropriate places that portray different objects. In this particular application, the usability of these filters are made even more difficult by the fact that there are very small patterns present on the floor and the constant motion the camera.

The last filter to be considered is one related to the codec induced artefact, which has resulted in the colour bands, as well as the grid pattern shown during earlier experiments. The Moving Pictures Experts Group codec used by the cameras perform a compression algorithm where the image is broken up into small squares, then are simplified as a combination of simple patterns before being heavily compressed by removing the insignificant patterns, especially the more complex ones. Because this process operates independently of the other squares, the continuity between the squares are lost and the inter square subtleties are often lost. Fig. 7 is a zoomed image of a carpet, which illustrates the block formation with a size of 4 by 4 pixels. After some experiments, it was observed that the compression factor, which contributed to how much of each square is smoothed, increased with higher capture resolutions to maintain the data rate going between the device and the processor.

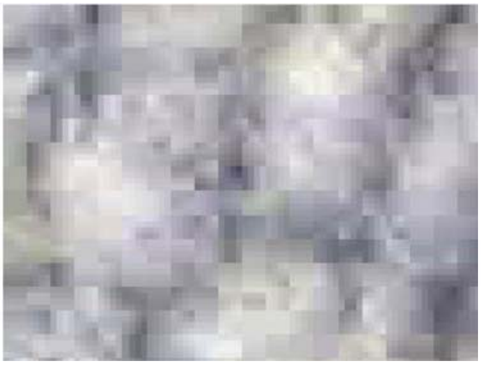


Fig. 7. Zoomed image of the carpet, the blocks from the codec is clearly visible

Instead of using a large spatial filter to remove the blocks from showing, like a Gaussian blur filter, a carefully weighted custom filter has been developed to blend between the squares, as well as smoothing between the inner 2 by 2 blocks that have formed. The weighting used between the pixels were chosen by observation, but manages to remove the majority of the grid formation, as shown in fig. 8. The weights that are shown are used in the following algorithm to evaluate the new intensity, where the number for the thick blue line is *A*, the number for the medium line is *B*, and *I* represents the intensity:

$$L = A, \text{ if } i\%4 = 0;\ B, \text{ if } i\%4 = 2;\ 0, \text{ otherwise}$$
$$R = A, \text{ if } i\%4 = 3;\ B, \text{ if } i\%4 = 1;\ 0, \text{ otherwise}$$
$$U = A, \text{ if } j\%4 = 0;\ B, \text{ if } j\%4 = 2;\ 0, \text{ otherwise}$$
$$D = A, \text{ if } j\%4 = 3;\ B, \text{ if } j\%4 = 1;\ 0, \text{ otherwise}$$
$$I_{i,j} = (L * I_{i-1,j} + R * I_{i+1,j} + U * I_{i,j-1} + D * I_{i,j+1} + 4 * I_{i,j}) \,/\, (4 + L + R + U + D) \qquad (1)$$

The current implementation makes use of only one of the above filters, which is the block removal filter with a weight of 1 and 0.25 for *A* and *B* respectively.

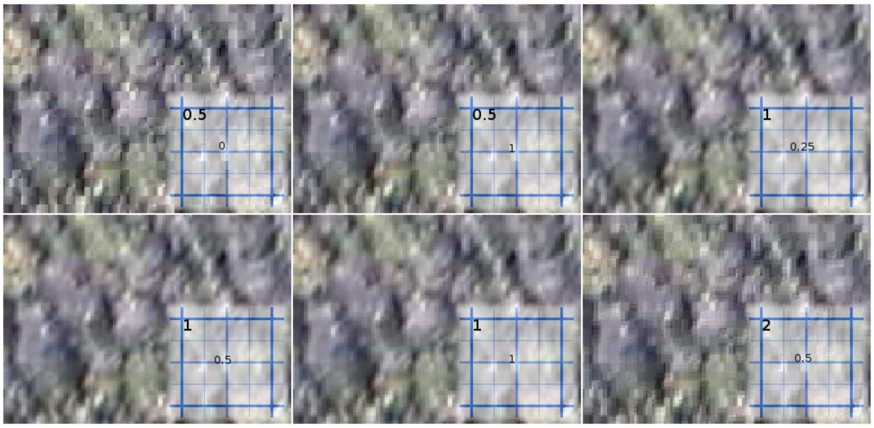


Fig. 8. Block formation removal, the numbers on the line represents the corresponding weight for the interpolation

4. Floor feature tracking

4.1 Feature detection

Although the area feature tracking is a well studied, many of the attributes depend greatly on the specific domains, thus it is important to incorporate the extra criterion or constraint into the feature tracking process.

One of the key considerations to make for this application is the lifetime of the features, as they are only required in the immediately subsequent frame. It is possible to store these features for a longer period in case the robot travels slower than expected, to correct any ambiguous tracking of features, or to capture a landmark for the purpose of global localization. However, maintaining the extra tracker increases the search area, reduces the speed of the robot, as well as introducing more processing tasks for the system. This constraint means that a new feature candidate is required at each cycle, which places a greater emphasis on quickly identifying a unique feature.

To determine the effectiveness of the features, a score must be evaluated for the candidates to select the best feature within the current view. An important criteria for the feature is to be able to distinguish the same set of pattern in the subsequent frame, thus must consist of a uniqueness measure as well as a consistency measure, which means the intensity pattern should not be too difficult to distinguish if the pattern appears slightly differently due to blurring or sub-pixel motion. Since the change in the appearance is kept small due to the short interval between the frames, more emphasis is required for the uniqueness factor as the floor texture often contains repeated patterns and lacks the variety of intensities found in scenes of other image processing scenarios.

Using the intensity value, it is possible to determine the fluctuation from some set value, such as the standard deviation score, to determine how different the region is compared to the surroundings. The uniqueness score only needs to be with respect to the immediate surroundings, as they are only maintained for one cycle. This means the average value should also be derived dynamically. The region being considered can range from the feature itself, the whole region being considered for the candidates, or the viewing area including regions that may or may not be traversed in the future.

A much simpler way to score is a simple accumulation of the pixel intensities within the region of interest. This allows the feature to be summarised with just one parse and can easily identify the brightest or the darkest region. However, this can thus lead to reduced reliability in terms of lack of consistency over time due to noise and the amount of similarity there is with respect to the surroundings.

An alternate way to evaluate the score is to use the difference in the intensity with the neighbours, such that the edge image is used instead of the intensity. Since the boundary is shared between two pixels, the edge score should not be considered on a per-pixel basis, but at a per-boundary basis. As the intensity difference scores are repeatedly used, a second image can be formed consisting of the change in the intensities. Once the edge map is derived, it can be re-used for other image processing tasks.

The candidates that are considered to be the feature should avoid bad patches of flooring that are not distinctive enough while being mindful of the processing load. Since the pattern on the floor cannot be pre-determined, the scanning sequence for candidates is usually set to a raster pattern. Using this sequence, the change in the score can be evaluated by observing the difference between the previous candidate and the current candidate. This is made possible by the fact that the scores for each pixel, or the boundary, are independent of each other. With this in mind, the scan only needs to consider three cases, as illustrated in fig. 9, to sequentially determine the score of the current candidate.

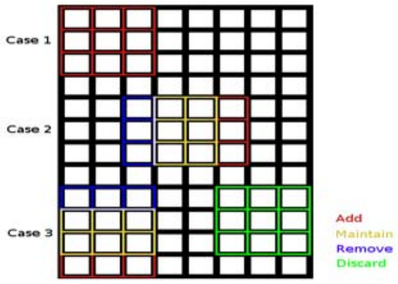


Fig. 9. Difference in the score, only the regions that enter and leave the window are updated

The approach showed a speed up from 3.21 ms to 2.04 ms when performing the standard deviation algorithm using a 16 by 16 grid for the feature size in an search area of 8 by 8 positions. The time was determined from 500 cycles of feature candidate selection.
As eluded to earlier, the uniqueness score is the most important aspect of the feature selection. Evaluating the effectiveness of this score can be conducted by observing the distribution of the score across the potential range and also on the distance between the adjacent scores. Table 1 summarises the results using the same condition as the above experiment, where the uniqueness is the average percentage rank in terms of finding the feature with the greatest difference in scores with the neighbours.

Algorithm	**Carpet**		**Vinyl**	
	Time (ms)	**Uniq. (%)**	**Time (ms)**	**Uniq. (%)**
max(ΣI)	2.33	91.23	2.27	97.45
min(ΣI)	2.31	73.23	2.41	89.06
$\max(\Sigma\|I_{x,y} - I_{ave\ (all)}\|)$	3.9	99.67	3.63	97.54
$\max(\Sigma\|I_{x,y} - I_{ave\ (view)}\|)$	3.81	96.66	2.61	95.55
$\max(\Sigma\|I_{x,y} - I_{ave\ (feature)}\|)$	4.09	97.1	3.01	88.05
$\max(\Sigma\|I_{x,y} - I_{x+1,y}\| + \|I_{x,y} - I_{x,y+1}\|)$	3.37	65.58	3.85	37.12

Table 1. Performance of scoring algorithms, the uniqueness score differed greatly with different implementations

It was interesting to observe that the uniqueness of a brighter region was more distinctive than darker regions. This may be due to the level of noise encountered at various intensities or the size of the bright and dark regions. As expected, the naïve approaches were not au very successful in identifying unique regions. Interestingly, the approaches using the difference in the intensities also did not perform well due most likely to the repetitive nature of the floor texture.

Out of the three standard deviation approaches, the one including the surrounding regions performed slightly better than the others. The algorithm performed well on both surface types that were tested, while the processing time did not differ too greatly.
When observing the ranking of the scores, it was noted that the majority of the best candidates were either the highest or lowest scoring candidate, mostly due to the fact that only one distance is used. One enhancement which was made is to evaluate not only the maximum, but to evaluate the top two and bottom two before selecting the one with the greatest difference to the second highest or lowest score.
The last of the issues to be considered is the shape and size of the features that are tracked. While the size is generally dependent on the processing capability and the amount of uniqueness that is present on the floor, the shape can be used to exploit any known configuration of the texture pattern to improve the effectiveness of the feature. To investigate the effects of different shapes and sizes, several arrangements were constructed.
Some of the surfaces that the robot could travel on contains streaks rather than random or spotty patterns. By using a rectangular shape that is aligned with the streak, it increases the chance of seeing the ends of the streak while minimising the processing of the uniform region between the streaks. Another arrangement that is used is a square, but with the central portion being removed. This arrangement was considered after observing that the carpet texture contained regions where the intensity did not differ greatly, such as the portion at the tip of each bundle of threads. The last arrangement is where the pixels of interest are separated and form a sparse grid formation. This arrangement allows a wider area to be covered with a small number of pixels being considered. Fig. 10 illustrates the various sizes and shapes that were considered as the feature.

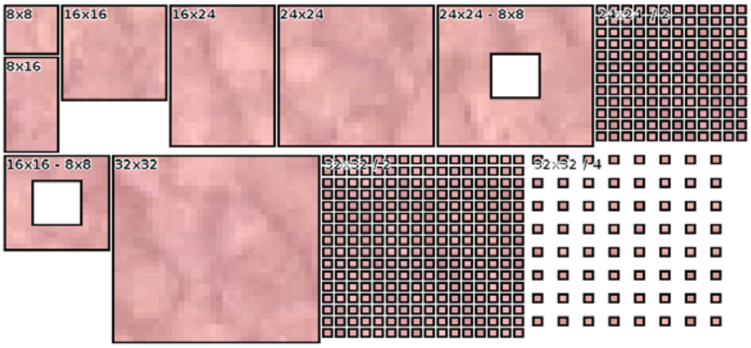


Fig. 10. Feature shapes, the red region represents the viewing area of the ground

The experiment was conducted in a similar way to the earlier tests, but the number of candidates was increased to 256. The scoring algorithm that was used was the standard deviation algorithm across the viewable area. The results, as shown in table 2, indicate the dramatic effect in the processing time when the number of pixels being considered increased, while the uniqueness score was quite high for the majority of the cases. One of the reasons for the high uniqueness score is from deciding between the maximum or minimum score, as described earlier, when selecting the feature.

Algorithm	Carpet		Vinyl	
	Time (ms)	Uniq. (%)	Time (ms)	Uniq. (%)
8x8 square	2.98	98.07	5.78	92.23
16x16 square	5.63	99.65	4.05	98.97
24x24 square	18.06	99.84	17.39	95.73
32x32 square	32.95	99.79	30.07	99.43
16x8 rectangle	3.13	99.54	5.92	92.58
24x16 rectangle	7.86	98.78	5.62	99.18
16x16 – 8X8 donut	3.09	99.7	7.56	99.68
24x24 – 8X8 donut	17.03	99.65	16.68	85.59
24x24/2 spaced	3.32	62.5	3.96	92.76
32x32/2 spaced	6.97	99.72	6.37	99.3
32x32/4 spaced	2.81	99.86	3.12	97.94

Table 2. Comparison of shapes, the square performed just as well as customised shapes

Although the uniqueness is not the highest, the 16 by 16 square was chosen as the feature dimension, due to the potential speed-up. The average intensity is derived during the image buffer copying process, thus saving some computation in traversing the image again.

4.2 Feature tracking

The basic sequence of tracking requires iterating over all candidate locations and determining the difference in the appearance with the stored feature. Due to the limited processing time, this operation must be carried out as efficiently as possible. Given that the tracking is of physical objects, their motions will be continuous. Even when the motion is represented using discrete amounts, the smoothness between the frames can be used to predict the location of the feature in the current frame (Kalman, 1960).

The basis of the motion prediction uses the motion vector from the previous cycle and appends the acceleration and the current motor command it is executing. This simple model can be evaluated very quickly, but relies heavily on accuracy of the previous motion that was determined. Since the motion prediction only guides the search area, as long as the prediction is close to the actual position of the feature, the correlation process will take care of correcting the error in the prediction. Determining the motion vector is as simple as using the previous motion that was determined after the feature was found. The acceleration is derived by taking the difference between the motion vectors, but does not consider the motor's characteristics in the current implementation.

The motion command to the motors is buffered due to the slight delay in executing and completing the motion. Given that the motion vector and acceleration already captures the majority of the robot motion, the motor command is currently only used to assist when the robot starts moving from a stationary position. The initial acceleration of the robot is very small, thus the magnitude of this adjustment is currently set to be only one pixel.

To verify the accuracy of the motion prediction, the error in the distance between the prediction and the correlated position was plotted, as shown in fig. 11. The experiment was carried out with the robot moving semi-randomly on a carpet surface for a period of approximately 2 minutes with a search area of 16 by 16 pixels. The types of motion included accelerating to full velocity, turning, reversing directions, and others which tested the extreme conditions in which the robot would operate. The results show that the prediction was quite close to the actual feature position with one very interesting result, which showed that a single pixel error was more common than a perfect match. This may be due to the acceleration always being delayed by one cycle or the frequent occurrences of sub-pixel motion. The plot also gives an idea as to how big the search area must be, since no match was found with a distance greater than 10 pixels away from the prediction, it is possible to reduce the search area as long as the types of motion for the robot has been fully explored.

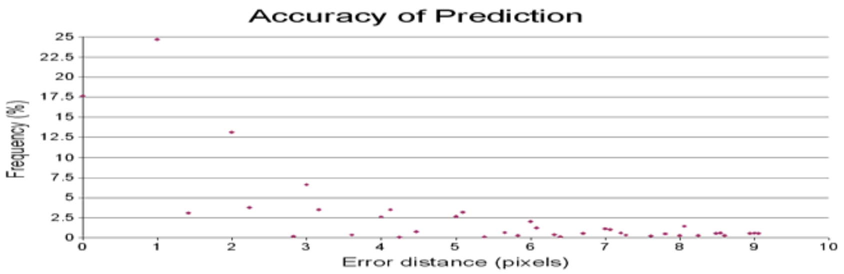
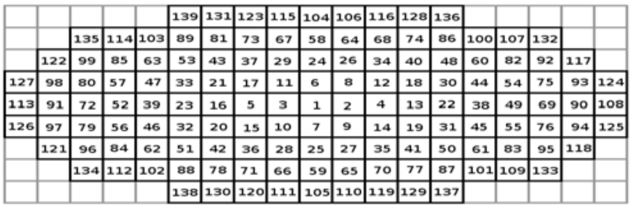


Fig. 11. Accuracy of prediction, the algorithm predicts the motion quite well

Given that the motion prediction is quite effective in finding the approximate location of the feature, it would be beneficial to make use of this when searching for the feature. A simple optimisation that can be implemented is by setting a threshold score that halts the computation of the correlation score if a better solution has already been found. Using a raster scan pattern to search for the feature does not fully utilise the sequence in which the correlation is carried out, thus a spiral scan pattern is introduced.
Given the higher probability in finding the ideal match near the predicted location, the scan pattern can be made circular, such that it forms a spiral shape as more candidates are considered. Since the ideal candidate is likely to have been found early on in the search, the threshold can quickly eliminate bad correlations. Another improvement that can be made is to acknowledge the axis in which the errors more frequently occur. Assuming that the camera is mounted to the side of the robot, the dominant motion the camera will observe will be along one axis, since the locomotive configuration will forbid lateral motion. This leads to more prediction errors occurring in one axis over another, which can be accounted for in the search area by shaping the spiral into an elliptical shape, as shown in fig. 12.



					139	131	123	115	104	106	116	128	136					
		135	114	103	89	81	73	67	58	64	68	74	86	100	107	132		
	122	99	85	63	53	43	37	29	24	26	34	40	48	60	82	92	117	
127	98	80	57	47	33	21	17	11	6	8	12	18	30	44	54	75	93	124
113	91	72	52	39	23	16	5	3	1	2	4	13	22	38	49	69	90	108
126	97	79	56	46	32	20	15	10	7	9	14	19	31	45	55	76	94	125
	121	96	84	62	51	42	36	28	25	27	35	41	50	61	83	95	118	
		134	112	102	88	78	71	66	59	65	70	77	87	101	109	133		
					138	130	120	111	105	110	119	129	137					

Fig. 12. Scan sequence, the positions are traversed in a spiral sequence

The overheads in computing the index offsets for the new sequence can hinder the performance of the algorithm, thus the relative offsets from the predicted location is computed beforehand, which only requires a small memory footprint.
To compare the performance of the new scan pattern, the spiral scan pattern was compared against a simple raster scan pattern of an equivalent search size, a 15 by 9 grid. The linear scan improved from 7.866 ms to 6.964 ms by using the threshold, while the spiral scan improved from 10.166 ms to 5.624 ms, which illustrate the large overhead in modifying the indexes for the scan pattern.
It is worth noting that the performance of the threshold optimisation is dependant on the surface textures, as the fluctuation in the correlation scores determines how quickly the bad candidates can be eliminated. The results above were taken on a vinyl floor, as it is one of the more common types of floors and consists of less distinctive patterns.
One of the biggest issues with precise tracking is the conversion of smooth motion to discrete amounts, as the precision is limited by the proportional size of each pixel. It is unlikely that the motion would align perfectly with the capture resolution characteristics, thus the small amounts of sub-pixel motion errors can accumulate to a large amount over time. Although the sub-pixel motions will eventually accumulate over time to result in a full pixel motion, the tracking algorithm cannot distinguish the amount of accumulated sub-pixel motions since the feature is only tracked for a single cycle. Given that the intensity of an object blurs and blends with the neighbour when the object is transitioning across adjacent pixels, the reverse can be simulated in software for the feature to generate how it may appear if it was viewed after sub-pixel motion.
A strategy to determine if sub-pixel motion occurred is by assuming that the overall correlation score are high in adjacent positions due to the size of the objects. By ranking the best scoring candidates, it is easy to determine if the neighbours of the best match is also a good candidate. This information can be used to combine the positions of the high ranking candidates to derive at a new position for the feature. Deriving the weight for the merger can be fixed or scaled based on the correlation scores.
An alternate way is to simply merge the neighbours and increase the number of candidates to match against. It is easy to make use of a fixed weight for this strategy, but to modify the weights, the scores of the neighbours of the best candidate must be evaluated. Using the threshold optimisation does not guarantee this, thus the neighbour's scores must be evaluated separately. The last strategy that was included was to apply the above idea to the predicted location. The idea behind this is to help recover against repetitive floor texture which can match an incorrect candidate due to large amount of blurring.
Testing for the above sub-pixel motion algorithms were carried out using two resolutions to determine the effects given a sample floor texture, which was a carpet floor. To maintain the

similarity between the two resolutions, the speed of the robot was reduced instead of increasing the search area. The testing consisted of moving the robot in a straight line for 2 meters and determining the difference between the measured motion and the actual motion. The results, as shown in table 3 indicate a dramatic improvement by using the sub-pixel motion algorithms, especially the combined approach which generates extra candidates around the best candidate using the correlation scores as the ratio if the neighbour has the second highest correlation score or if the second and third best candidates are neighbours of the top, but in perpendicular directions with each other.

Algorithm	**Error (%)**	
	Resolution	
	160 x 120	**320 x 240**
None	5.5	1.93
Fixed weight with neighbour of best candidate	5.2	1.67
Variable weight with neighbour of best candidate	1.02	0.32
Fixed weight merge of best neighbouring candidates	1.43	1.41
Fixed weight with neighbours of prediction	3.32	1.5
Combined	0.98	0.28

Table 3. Sub-pixel motion detection, the improvements from the algorithm can be seen on both resolutions

Since an undetected sub-pixel motion is lost and a new feature is determined with each frame, it is important to override this by maintaining the same feature for multiple frames if it can. Since the assumption forbids rotations from occurring and the robot does not allow the same feature to be within the view if it is travelling faster than half its maximum velocity, keeping the same feature for consecutive frames should only occur if the robot is travelling very slowly. To simplify this case, the feature is not discarded if the localization algorithm does not detect any motion, thus handling the issue of detecting very subtle motions when the robot stops and starts moving.

5. Multi-sensor localization

5.1 Multiple tracker

The feature tracker introduced above performs accurately and reliably, but is limited to detecting displacement. This means any accumulation of motion that is evaluated is invalidated as soon as there is a change in the orientation. To be able to determine if rotation has occurred, another constraint must be introduced to determine how much rotation has occurred.

The first approach is to constrain the type of motion to either pure translation or pure rotation and placing the camera in a configuration such that the two types of motion can be distinguished and measured. Given the configuration of the wheels, which are in line with the center of the robot, the motions can be controlled such that the magnitude of the wheel speeds are always in sync to only allow a pure translation or rotation. Assuming that the tracker is placed at the front or the back of the robot, if a translation occurs, the tracker will detect motion along the same axis, while a rotation will result in the majority of the motion occurring in the perpendicular axis with a small amount of motion in the other. Fig. 13 illustrates the direction of the motions detectable by the camera at various positions around the robot.

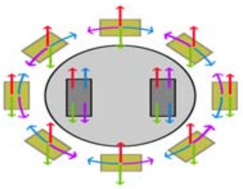


Fig. 13. Motion direction for different wheel motion, the colours correspond to the detected direction of motion

The second approach is to use multiple trackers with known positions, where the motion detected by both trackers combined with the placement of the trackers provide the extra constraint to be able to determine the robot motion. Using this approach can allow for a wider range of motions to be detected, but effectively doubles the processing load as they must operate completely independently. The placement of the second tracker should ideally be located as far away from the first tracker as possible, as this would allow different types of motions to be determined more easily given the issue with precision of the individual motions. Two strategies were considered for this, which were to use a single camera and place the trackers as far apart as possible, or to use two cameras and freely control where they would be placed.
Initial testing of the single camera approach showed appalling results for both the single tracker approach with the camera at the back of the robot, at 53.16% and 62.43% error for translation and rotation respectively, and the two tracker approach where the camera was placed on the side, at 2.05% and 69.2% error for translation and rotation respectively. The tests were carried out by moving the robot forwards for 1 meter, then reversing back to the original location for the transition test. The rotation test was done by spinning the robot on the spot for 360 degrees, then reversing back to the original location. The algorithm used for the single tracker approach was set such that the decision between translation and rotation was made by the higher magnitude in the two axes of motion. This naïve approach did not perform well due to the unrealistic assumption which prohibited any rotation other than at the center of the robot, as well as the clear distinction between translation and rotation. As for the multiple tracker approach using a single camera, the small distance between the trackers meant the sensitivity of each of the trackers were extremely high, thus contributing to the errors when they were out of sync.
The approach involving multiple camera involves several considerations before it can be used for the localization algorithm. The first issue arises from being able to access the camera, as older device libraries on some operating systems prohibit the simultaneous use of devices of the same type. This has been rectified in later versions of libraries, but some device libraries still does not allow multiple cameras being used simultaneously. Since the area within the frame that is used is quite small, it is possible to manufacture a physical device with mirrors to view the ground at various positions around the robot using one

camera, but this is not considered at this stage due to the production costs and potential issues with adjustments later on.

A related issue to note is the bandwidth limitation of the buses which the cameras are connected to via USB ports. The device driver will tend to favour one of the devices if they share the same bus, which results in a full frame rate for one camera and a reduced frame rate for the other. A simple solution to this is to make sure the cameras are connected on different buses, so that they both run at the full frame rate.

Since the two trackers are captured separately, it is important to be able to synchronise the two devices, such that the motions are captured for the same time period. To measure the timing between the two cameras, both devices were exposed to a sudden change in the lighting condition, which was achieved by toggling the room's light. After numerous test runs, both cameras were observed as capturing the change in the light condition at the same time, but slightly delayed from the perspective of the light being toggled and the observer. At this point in time, the latency in the reaction time is not a significant issue due to the lack of synchronism with the other modules of the mobile robot system. The lack of difference between the capture time for the two camera devices meant the information could be combined without the need to buffer and interpolate the captured motions.

Since the cameras can be placed freely around the robot, the placement of the cameras is also something that is considered. However, many of the places around the robot can simply be eliminated for consideration after some careful study of the consequences. Since the typical motions of the robot is already known, aligning the axes with the camera allows for constrained or reduced sub-pixel motions. This eliminates all but four positions; the front, the back, and two sides.

By placing the camera at the front, it adds an extra benefit of being able to observe low objects that have been missed by the range finders or sudden change in the floor continuity, such as stairs. However, since these break the assumption made to the localization approach, they are not critical issues at this stage. Even if these hazards existed, there are better sensors that can be used to detect these, such as a range finder pointing downwards. Placing the camera at the front introduces the risk of potentially colliding into something with the camera if the range finders are not able to detect the obstacles, such as if the camera mount blocks the view of the range finders. Placing the camera at the back provides similar benefits to the front in terms of being able to easily distinguish translation and rotation, but is generally a lot safer as the robot typically travels forwards.

Placing the camera to the side means the majority of motion will be along the same axis, thus reducing the search area. One of the benefits of this approach over the front is that since the robot travels forwards, the range finder adjacent to the camera mount can inform the robot if there is an obstacle that is about to collide with the camera. Although using a single camera on the side was unable to distinguish between translation and rotation, the additional camera will be able to provide the extra information as to which type of motion is occurring.

With the above in mind, three different arrangements were tested; back and side, back and side with dedicated axis, and two side cameras. The first and the third configurations were carried out in the similar fashion to the single camera experiment, while the second configuration attempted to assign a dedicated role to each of the camera. This was achieved by giving the role of rotation checks to the back camera and translation checks to the side camera. If the back camera did not observe a sidewards motion above an arbitrary selected

threshold amount of 2 pixels, the side camera was used to determine how much translation occurred. Note that this arrangement was for comparison purpose only, as it does not give the freedom in movement like the other arrangements.
Once again, the results did not show anything plausible to be used as a localisation algorithm with the back-side arrangement performing at 22.55% and 41.99% error, 36.84% and 32.68% error for the dedicated tracker for each axis, and 2.88% and 82.12% error for the side-side arrangement, where the first percentage value is for translation and the second is for rotation. However, it did indicate some improvement and consistency in the measurements, with the exception of the rotation for the two sided configuration. During the experiment, it was noted that although the frames were being captured at the same time, the two motions that were captured were not properly synchronised in terms of magnitude and orientation. This was due to the sub-pixel motions that were not detected at the same time, thus leading to invalidating assumptions made by the use of the motion models.

5.2 Synchronisation

The subtle errors in the two motions meant any constraint placed between the trackers is invalid due to the incorrect information being combined. To resolve this issue, two strategies are introduced in the attempt to synchronise the two motion vectors. The first strategy involves smoothing the motion vectors, such that the sub-pixel motions are gradually accounted for. To achieve this, two basic approaches are considered, which are accumulation and windowing approaches.
The accumulation involves the idea of summing the motion vectors for a number of cycles before the motion of the robot is calculated. One of the issues with this approach, as shown in fig. 14. is the misalignment of the resulting orientation as the sequence information is lost. This suggests that the more motion vectors that are buffered, the more opportunity there are for misalignment to occur.

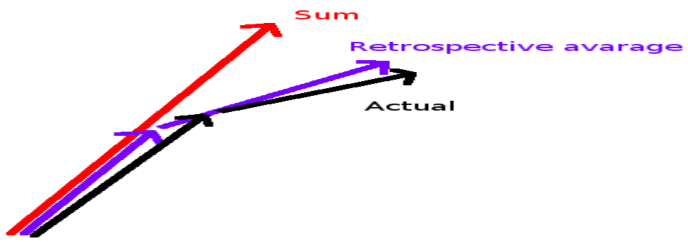


Fig. 14. Error from merging the motion vectors, the sequence information is lost after the vectors are merged

The results in table 4 compares the various buffer sizes using a side-side camera configuration and a differential motion model, which will be described later. Although the trend is not as obvious, it has performed better for translation and worse for rotation with increased buffer size, which makes it difficult to set the ideal buffer size.

Buffer size	Translation (%)		Rotation (%)	
	Forward	Backward	Clockwise	Anti-clockwise
4	3.21	0.58	11.43	1.22
8	4.1	0.59	10.66	0.87
16	3.57	0.3	13	2.76
32	3.83	0.29	10.9	5.2

Table 4. Performance using different buffer size, the large buffer leads to better translation but worse rotation performance

Since the error in the rotation occurs when the robot is not travelling straight, it is possible to modify the triggering condition for emptying the buffer to encourage the use of the buffer when travelling straight, while discouraging motion vectors of varying orientation being accumulated. Table 5 shows four different approaches that were tested, which did not seem to show much change to the previous implementation when ignoring scale based error correction.

Trigger condition	Distribution	Translation (%)		Rotation (%)	
		Forward	Backward	Clockwise	Anti-clockwise
Different quadrant	One	3.71	0.95	12.51	0.72
Different quadrant or neutral	One	4.22	0.68	12.6	0.81
Different quadrant, neutral, or buffer size is reached	One	3.5	0.44	11.38	0.65
	Average	5.67	0.91	9.28	0.58

Table 5. Trigger conditions for emptying the buffer, the performance were mostly better than the fixed buffer size approach

The second strategy in smoothing the motion involves the use of a shifting window on a buffer of recent motion vectors to blend the portions of the of vectors together. Each motion vector is thus divided into smaller portions, where the sum of the weights used equals 1. Deciding how to distribute the weights involves observing the effects of splitting the motion vector, which is shown in fig. 15. Note that the error plot on the right should have an error value of zero for weight of zero, as no rotation would occur for the initial motion. Similarly to the accumulation approach, the change in the sequence of the motion results in an incorrect pose being maintained. As the diagram shows, it is better to apply a larger weighting early on than vice versa, which also allows for the immediacy in the motion being recognised.
To test the above theory, several weight functions were considered, as shown in fig. 16. As well as the window size, some of the functions required another factor to determine the amount of distribution, thus some variation of this were also tested. The results in table 6 show comparable performance to the accumulation approach, especially those with a smaller window size. Note that the normal distribution does not add up to 1, thus end up falling short of the actual motion, which explains the poor performance.

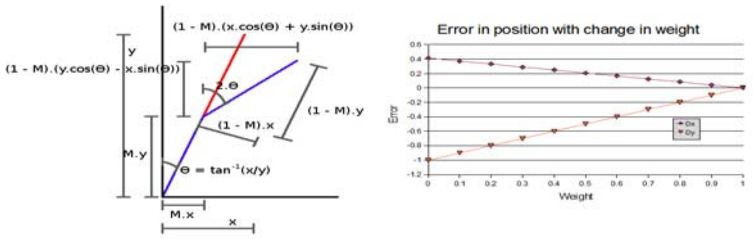

(1 - M).(x.cos(Θ) + y.sin(Θ))
y
(1 - M).(y.cos(Θ) - x.sin(Θ))
2.Θ
(1 - M).x
(1 - M).y
M.y
Θ = tan⁻¹(x/y)
M.x
x
Error in position with change in weight
Error
Weight
Dx
Dy

Fig. 15. Splitting the motion vector, the components are shown on the left and a sample error plot is shown on the right

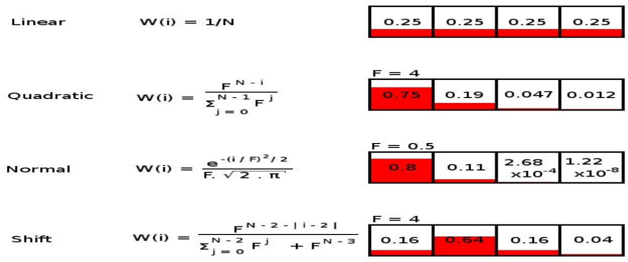


Fig. 16. Weight functions for smoothing, sample distribution is shown on the right

Distribution	**Factor**	**Size**	**Translation (%)**		**Rotation (%)**	
			Forward	**Backward**	**Clockwise**	**Anti-clockwise**
Constant	1	4	3.1	0.86	10.87	1.37
		8	3.98	0.88	11.65	1.08
Quadratic	2	4	5.23	1.03	12.2	2.01
		8	4.8	0.89	11.9	1.34
		16	4.91	0.91	12.42	1.27
	4	4	2.88	0.78	9.72	0.97
		8	2.91	0.87	9.84	0.89
		16	3.01	0.8	11.21	1.14
Normal	0.5	4	2.94	0.89	10.02	0.98
		8	2.91	0.79	8.78	1.03
	2	4	37.65	1.31	64.84	12.73
		8	24.36	1.16	43.64	9.05
Shifted	4	4	2.95	0.92	10.87	0.88

Table 6. Performance of different weight functions, the factor and size both play an important role in the performance

Since the immediacy in the motion detection allows for the system to react much faster, as well as being simpler to synchronise with the other modules, the windowed approach with a quadratic weight distribution of size 4 was selected as the smoothing algorithm.

5.3 Motion model

As eluded to earlier, the use of a different motion model to combine the two motion vectors allows for a more error tolerant system due to extra constraints placed on the types of motion that are allowed. However, the added constraint means the usage must be carefully considered and matched up with the type of motions that are expected. The first model to be considered in detail is the exact motion model, which has been used throughout the earlier experiments, while the second model, the differential motion model, was used for the synchronisation section.
The common idea behind both the motion models involves the derivation of a rotational point and an angle to describe the instantaneous motion, with the exception being a pure translation motion, which is simply a displacement with no change in the orientation. The pivot point, which is called the ICC (Instantaneous Center of Curvature) is derived differently for the two motion models, where the exact motion model allows this point to be anywhere, while the differential motion model requires that the ICC remain on the same axis as the rotational axis of the wheels.
Using the exact motion model involves the idea of splitting the motion vector into half, where the perpendicular line from the mid-point of the motion vector intersects at the ICC. By having the two motion vectors at different angles, the intersection of the two lines can be determined. If the slope of the two lines are equal, the robot is said to be in a pure translation. Fig. 17 illustrates the components involved in using the exact motion model.

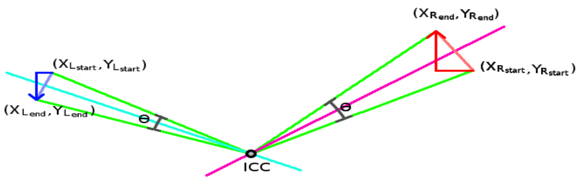

$(X_{R_{end}}, Y_{R_{end}})$
$(X_{L_{start}}, Y_{L_{start}})$
$(X_{R_{start}}, Y_{R_{start}})$
$(X_{L_{end}}, Y_{L_{end}})$
θ
θ
ICC

Fig. 17. Exact motion model, the bisector of motion vector intersects at the ICC

It should be apparent that the inaccuracy in the motion vectors can lead to significantly different ICC and rotational angle being derived, which explains the poor results from earlier experiments.
The differential motion model is a more lenient approach, which relies on knowing the locomotive characteristics and the types of motions that are allowed with it. Fig. 18 illustrates the components involved, as well as the the derivation of the ICC and the rotational angle Since the positions of the two trackers are known, it is a simple process to derive the missing values. As the diagram suggests, the placement of the cameras should be aligned with the wheels to take advantage of the constraint on the location of the ICC.

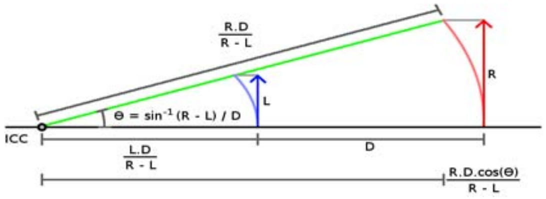


Fig. 18. Differential motion model, the ICC is constrained to be along one axis

By using the differential motion model, the localization algorithm assumes perfect behaviour by the locomotive components, as well as the lack of external forces that may force the robot to an unnatural motion. Although the effects of these types of motion will be small, it is worth while to consider an enhancement on the approach to handle the special cases.

The core of the hybrid motion model is the differential motion model, which is able to perform reasonably well on its own. The case which trips the differential motion model is when the motion vector of the camera does not fall within the allowed range, where the range is determined by the velocity of the two motions and expected motion in the perpendicular direction, as shown in Fig. 19. Using this plot, it is possible to determine the expected perpendicular motion for a pair of motion vectors. Although it is possible to derive this dynamically, a look-up table can also be used, since the number of entries is limited by the maximum magnitude of the motion vectors. When checking to see if the motion is within the bounds of the constrained motion or not, a small tolerance amount must be included to account for the sub-pixel motions and the latency introduced by the smoothing algorithm.

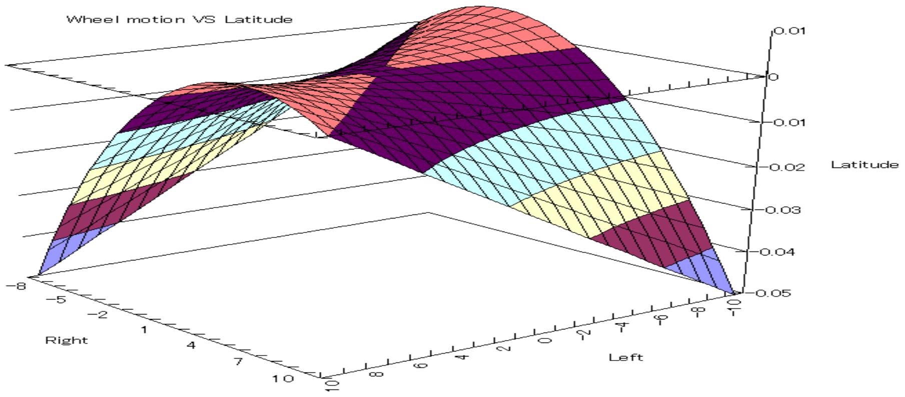


Fig. 19. Wheel motion against lateral motion, lateral motion is maximised when the robot spins at the center.

If the motion falls outside of the expected range, an alternate algorithm must be used to calculate the robot. The first approach which has been tested uses the exact motion model, while the second approach assumes that the error was caused by a bump of some sort and models the motion as a pure translation based on the average of the two motion vectors. The

result showed that using the exact algorithm produced an error value of 2.98% and 16.42% for translation and rotation respectively, while treating the motion as translation produced a result of 2.44% and 5.32% error for translation and rotation respectively.
A final comparison is made in table 7 with the three motion models using the windowed smoothing algorithm with a quadratic weight function of size 4 and factor of 4. All of the motion models used the two cameras on the side configuration, which provided the best performance out of the other arrangements. The results show a significant improvement over the naïve approaches and appear to be a promising localization algorithm. It is worth noting that the difference between the forward and reverse measurements is mostly due to incorrect scales being used, thus can be reduced by the calibration process in determining the height of the camera and the viewing angle.

Motion model	**Translation (%)**		**Rotation (%)**	
	Forward	**Backward**	**Clockwise**	**Anti-clockwise**
Exact	2.88	1.21	88.12	9.2
Differential	2.88	0.78	9.72	0.97
Hybrid motion	1.95	0.41	2.32	0.53

Table 7. Performance of hybrid motion model, the optimal algorithms for the other components are used for this test

6. Practical considerations

As briefly mentioned earlier, it is important to provide a precise calibration information in terms of the camera configuration, such that any scaling errors can be eliminated. This becomes a crucial issue when the robot is required to interact more with the environment, as the coordinate systems used to represent the pose of the objects within the environment will determine the actions the robot must perform
Other practical considerations to make is the type of environment the robot will be operating in. The current system is only designed to operate indoors, due to the physical configuration of the wheels, thus the performance of the proposed localization algorithm are tested on a wider range of indoor surfaces. A collection of surface types were found and tested on, which is summarised in table 8. The results are quite promising, except for some of the very dark and glossy surfaces. The similarity in the appearance make it difficult for the localization algorithm to correctly track the features, which resulted in the errors being introduced.
The final test that was conducted was based on the effects of the accumulation over a longer term period. Although the error rates show reasonable performance, the difference between the actual motion, the proposed algorithm, and the standard wheel encoder based dead reckoning approaches are compared for a traversal of the lab environment, as shown in Fig. 20. Since the algorithm derives the pose changes based on actual observations of movement, it provides a far better model of the robot motion as it bypasses many inaccuracies in modelling the locomotive characteristics. One major advantage of this approach is the level of accuracy that is maintained after a change in the orientation, which is the primary cause of long term misalignment.

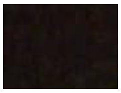
Surface	Sample	Translation (%)		Rotation (%)	
		Forward	Backward	Clockwise	Anti-clockwise
Vinyl		1.87	0.29	2.83	0.58
Table		2.37	0.52	3.08	0.92
Timber		8.21	1.07	7.18	3.84
Rubber		17.3	0.28	18.71	6.17
Tile		1.94	0.2	2.96	0.86
Brick		1.7	0.42	3.21	1.6
Concrete		2.44	0.27	2.69	0.76

Table 8. Performance on different surfaces, all but the dark textures performed well

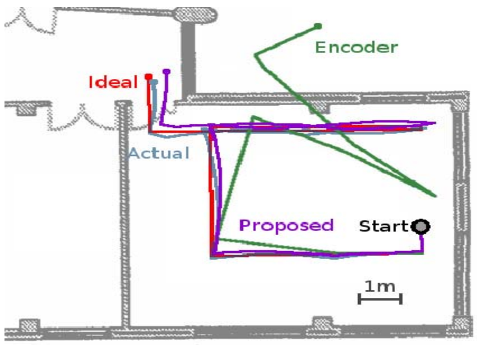


Fig. 20. Long traversal, the path's length was 26.5m in length

7. Summary

A local localization algorithm for mobile robots has been proposed, which is based on the idea of using multiple off-the-shelf webcams to perform ground texture tracking. The localization module has been developed on a custom built robot and tested in real indoor environments with dramatic improvement over encoder based dead reckoning approaches.
To take advantage of the constraints provided by the system and the type of environment the robot is exposed to, various characteristics of the camera were configured and adjusted to reduce the complexity in the tracking task. There are two constraints that are used for the proposed approach to work, which are:

- The elevation of the camera to the ground remains constant, and
- The features being tracked can only translate and not rotate in between frames.

Due to the processing requirement, only two filters are actively used, which are the lens warp removal filter and block removal filter. After exploring several scoring algorithms to find the feature, a simple algorithm based on the standard deviation has been used with a shape of 16 by 16 pixel square. To improve the processing time for finding the feature, a prediction is made to where the feature is located, followed by a spiral search sequence to quickly find the best candidate, which has lead to approximately 30% speed up.
By accounting for some of the sub-pixel motions by interpolating around the best candidate, the precision of the tracking increased by approximately 6 times.
To distinguish between translation and rotation of the robot, a second tracker was introduced to form a two-cameras-on-the-side configuration. The two motion vectors were smoothed by using a sliding window of size 4 and a quadratic weight decay function to better synchronise the two data sources. A hybrid motion model has been introduced to handle two types of motions; regular motion based on the locomotive constraints and irregular motion, caused by bumps and sudden slippages. By switching between the two, the performance of the algorithm showed some improvements even though the frequency of erroneous tracking is already quite small.
The proposed localization algorithm has been tested on various surfaces types that are commonly found in indoor environments with less than 1% error on both translation and rotation. It was found that the algorithm did not operate so well on very dark surfaces with highly repetitive or indistinguishable texture patterns. As long as the constraints can be maintained, the approach allows for an immediate and precise localization with low cost hardware at a reasonably small processing cost.

8. References

Davison, A.J. (1998), *Mobile Robot Navigation using Active Vision*, Thesis, University of Oxford.

Jensfelt, P. (2001), *Approaches to Mobile Robot Localization in Indoor Environments*, Thesis, Royal Institute of Technology.

Kalman, R.E. (1960), A New Approach to Linear Filtering and Prediction Problem, *Journal of Basic Engineering*, Vol. 82, Series D, pp. 35-45.

Krootjohn, S. (2007), *Video image processing using MPEG Technology for a mobile robot*, Thesis, Vanderbilt University.

Marchand, E. & Chaumette, F. (2005), Features tracking for visual servoing purpose, *Robotics and Autonomous Systems*, Vol. 52, No. 1, pp. 53-70.

Ng, T.W. (2003), The optical mouse as a two-dimensional displacement sensor, *Sensors and Actuators*, Vol. 107, No. 1, pp. 21-25.

Ritter, G.X. & Wilson, J.N. (1996), *Handbook of Computer Vision Algorithms in Image Algebra*, CRC press, United States.

Se, S.; Lowe, D. & Little, J. (2002), Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks, *International Journal of Robotics Research*, Vol. 21; Part 8, pp. 735-758.

Shi, J. & Tomasi, C. (1994), Good features to track, *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593-600.

Sim, R. & Dudek, G. (1998), Mobile robot localization from learned landmarks, *In Proceedings of IEEE/RSJ Conference on Intelligent Robots and Systems*, Vol. 2, pp. 1060-1065.

Thrun, S.; Fox, D.; Burgard, W. & Dellaert, F. (2001), Robust Monte Carlo Localization for Mobile Robots, *Artificial Intelligence*, Vol. 128, No. 1, pp. 99-141.

Vilmanis, Y. (2005), *Intelligent Robot System*, Thesis, Flinders University.

Wolf, J.; Burgard, W. & Burkhardt, H. (2002), Robust Vision-Based Localization for Mobile Robots using an Image Retrieval System Based on Invariant Features, *In Proc. of the IEEE International Conference on Robotics & Automation*, Vol. 1, pp. 359-365.

18

Omni-directional vision sensor for mobile robot navigation based on particle filter

Zuoliang Cao, Yanbin Li and Shenghua Ye
Tianjin University of Technology
P.R. China

1. Introduction

Automatic Guided Vehicles (AGV) provide automated material movement for a variety of industries including the automobile, chemicals/ plastics, hospital, newspaper, commercial print, paper, food & beverage, pharmaceutical, warehouse and distribution center, and manufacturing industries. And they can reduce labor and material costs while improving safety and reducing product and equipment damage.
An AGV consists of one or more computer controlled wheel based load carriers (normally battery powered) that runs on the plant floor (or if outdoors on a paved area) without the need for an onboard operator or driver. AGVS have defined paths or areas within which or over which they can go. Navigation is achieved by any one of several means, including following a path defined by buried inductive wires, surface mounted magnetic or optical strips, or alternatively by way of visual guidance. (Crisan D, &Doucet A. 2002).
This chapter describes a total navigation solution for mobile robots. It enables a mobile robot to efficiently localize itself and navigate in a large man-made environment, which can be indoor, outdoor or a combination of both. For instance, the inside of a house, an entire university campus or even a small city lie in the possibilities.
Traditionally, other sensors except cameras are used for robot navigation, like GPS and laser scanners. Because GPS needs a direct line of sight to the satellites, it cannot be used indoors or in narrow city centre streets, i.e. the very conditions we foresee in our application. Time-of-flight laser scanners are widely applicable, but are expensive and voluminous, even when the scanning field is restricted to a horizontal plane. The latter only yields a poor world representation, with the risk of not detecting essential obstacles such as table tops. (Belviken E, &Acklam PJ. 2001)
That is why we aim at a vision-only solution to navigation. Vision is, in comparison with these other sensors, much more informative. Moreover, cameras are quite compact and increasingly cheap. We observe also that many biological species, in particular migratory birds, use mainly their visual sensors for navigation. We chose to use an omni-directional camera as visual sensor, because of its wide field of view and thus rich content of the images acquired with. Besides, we added a few artificial markers to the environment for navigation.
In a word, we present a novel visual navigation system for the AGV. With an omni-

directional camera as sensor and a DSP as processor, this system is able to recover distortion of a whole image due to fish-eye lens. It can recognize and track man-made landmarks robustly in a complex natural environment, then localize itself using the landmarks at each moment. The localization information is sent to a computer inside the robot and enables fast and simple path planning towards a specified goal. We developed a real-time visual servo technique to steer the system along the computed path.
The whole chapter falls into seven sections. Section I introduces various navigation strategies widely used in the mobile robot, and discusses the advantages and disadvantages of these navigation strategies firstly. Then a vision-only solution to navigation is proposed, with an omni-directional camara as visual sensor. This visual navigation strategy can guide the mobile robot to move along a planned path in a structured environment.
Section II illuminates the framework of the whole system, including the targeted application of this research, an automated guided vehicle (AGV). The two-color sequence landmarks are originated to locate and navigate the AGV. Besides, a visual image processing system based on DSP has been developed. The entire related distortion correction, tracking and localization algorithm are run in the built-in image processing system. This compact on board unit can be easily integrated into a variety of mobile devices and appears low power, well modularity and mobility.
Section III lists all sorts of fish-eye lens distortion models and sets forth a method of distortion correction. The involved fish-eye lens satisfies isometric projection, and the optical imaging center and a distortion parameter of the visual sensor need to be figured out in order to realize distortion correction. Thus the classical calibration for common lens paramenters is applied to the fish-eye lens, and those above parameters are figured out.
Section IV talks about the novel real-time visual tracking with Particle Filter, which yields an efficient localization of the robot, with focus on man-made landmarks. It is the key technology that make the whole system work. The original tracking algorithm uses the result of object recognition to validate the output of Particle Filter, which improves the robustness of tracking algorithm in complex environment.
Section V puts stress on the localization and navigation algorithm with the coordinates of the landmarks provided by particle filter. The location and the orientation of the AGV are worked out based on coordinate transformation, in the three-dimensional enviromnent rebuilt on the two-color sequence landmarks. Moreover, a PID control strategy is run in the built-in computer of the AGV for navigation.
Section VI presents the actual effect of the mobile robot navigation. The experimental environment and the experimental steps are introduced in detail, and six pictures of experiment results are shown and discussed.
Secion VII summarizes the work done about the research. A beacon tracker based on Particle Filter is implemented in the built-in image processing system. Real-time distortion correction and tracking algorithms are performed in the system, and the AGV is located and navigated with the tracking results of hte landmarks from the beacon tracker.

2. System framework

Our mobile robot platform is shown in Fig. 1. With our method, the only additional hardware required is a fish-eye lens camera and an embedded hardware module. The fish-eye lens is fixed on the top of the vehicle to get omni-directional vision, and the embedded system based on DSP takes charge in distortion rectification, target recognition, target tracking and localization.

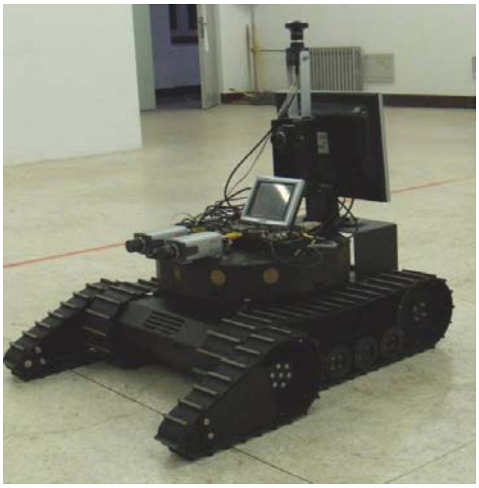


Fig. 1. Mobile robot platform

2.1 Mobile robot

Usually, a mobile robot is composed of the body, battery and charging system, drives, steering, precision parking device, motion controllers, communication devices, transfer system, and visual navigation subsystem and so on. The body includes the frame and the corresponding mechanical structures such as reduction gearbox, motors and wheels, etc, and it is a fundamental part of the AGV.

There are three working ways for an AGV:

(1) Automatic mode. When the AGV is set in automatic operation mode, the operator enters the appropriate command according to the plan path, and the AGV start to work in the unmanned mode;

(2) Semi-automatic mode. The operator can directly assist the AGV to complete its work through the buttons on the AGV;

(3) Manual mode. The operator can also use remote control trolley to move the AGV to the desired location manually.

The mobile robot platform of this chapter is a tracked AGV, and it consists of an embedded Industrial Personal Computer (IPC), motion control system, multiple infrared sensors and ultrasound sensors, network communication system and so on. The IPC uses industrial-grade embedded motherboard, including low-power, high-performance Pentium-M 1.8G CPU, SATA 160G HDD, DDR400 2G memory, six independent RS232 serial ports, eight separate USB2.0 interface. Moreover, four-channel real-time image acquisition card can be configured on this motherboard. The specific hardware modules of the mobile robot are shown in Fig. 2.

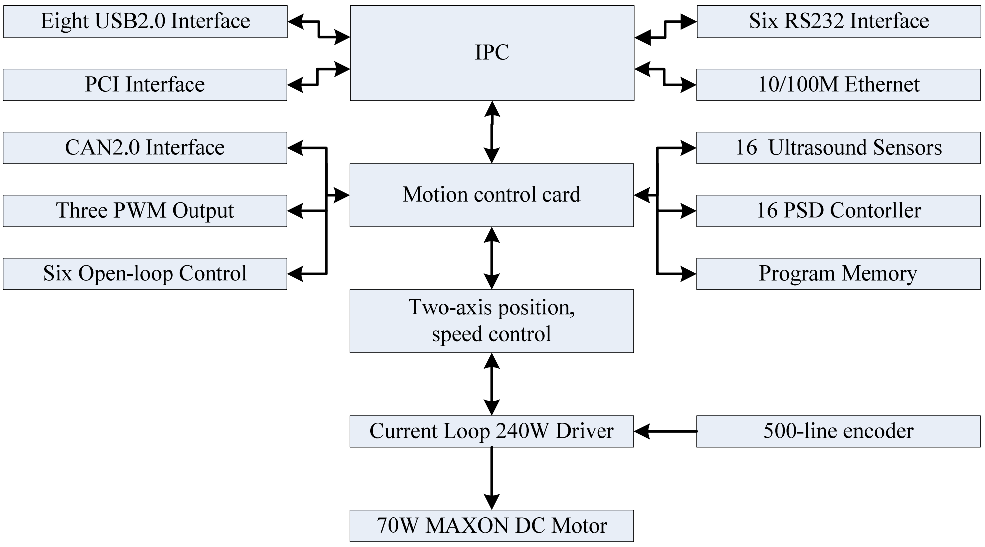


Fig. 2. Hardware modules of the mobile robot platform

2.2 Fish-eye lens

The project research needs high-quality visual image information, so that the camera features color, planar array, large-resolution and CCD light-sensitive device. We adopt imported Japanese ultra-wide angle fish-eye lens Fujinon FE185C046HA-1 as well as analog color CCD camera Watec221S, as shown in Fig. 3.

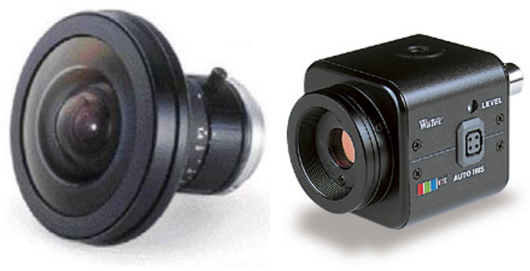


Fig. 3. Fish-eye lens camera

The performance parameters of the fish-eye lens are shown in Table 1. And the CCD size is 1/2 inches, PAL stardard, and the resolution (horizontal) is 50 lines (Y/C 480 lines). Its effective pixel (K) is P440K, minimum illumination is 0.1Lux, and lens mount method is CS installation, with operating voltage of DC +12V.

Focus(mm)	1.4
Aperture Range	F1.4-F16
CCD size	1/2”
Minimum object distance(m)	0.1
BFL(mm)	9.70
Interface	C
Weight(g)	150

Table 1. The performance parameters of the fish-eye lens

2.3 Embedded hardware platform

The vast majority of image processing systems used currently by the AGV are based on the traditional PC or high-performance IPC. Although the PC-based architecture is simple and mature technically, and applied widely, these image processing systems are redundant in terms of both resource allocation and volume, besides they have poor flexibility, heavy weight and high power consumption, which is not suitable for the mobile vehicle system application.

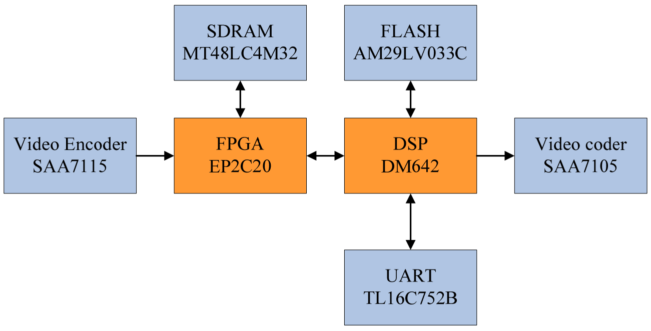


Fig. 4. System diagram of hardware platform

While the embedded system, as a highly-integrated application platform, features great practicability, low cost, small size, easy expansion and low power consumption. Therefore, we drew on the current successful application of DSP and FPGA chips in the multimedia processing, considering the characteristics of the vehicle image processing system such as large computation, high real-time requirement and limited resources, proposed a solution to build an embedded hardware image processor based on FPGA+DSP. And target recognition, tracking and localization are achieved on this hardware platform. The system diagram of the hardware platform is shown in Fig. 4.

We use Altera’s Cyclone series FPGA EP2C20 and TI’s DaVinci DSP TM320DM642 to build the embedded hardware image processor. Firstly, the input analog video signal goes

through clamp circuit, anti-aliasing filter, A/D conversion and YUV separation circuit to be converted to BT.656 video data stream. Secondly, the rich internal hardware resource of the FPGA is used to achieve image capture, color space conversion, image enhancement and distortion correction and so on. Thirdly, other advanced image processing algorithm such as target recognition, tracking and localization, are implemented in the DSP with its high-speed signal processing capability. Finally, the result of image processing is output to the SAA7105, which converts it into NTSC or PAL television signal and displays it on a liquid crystal screen. In a word, the entire system fully plays their respective advantages of the DSP and FPGA, and combines them closely to form a dedicated embedded hardware image processor.
In addition, we equip the hardware image processor with high-speed, large-capacity data memory and program memory, in order to meet the requirements of high-speed image processing and transmission.

2.4 Two-color sequential landmarks

The target recognition algorithm in this project belongs to the template matching method, and the invariant features of the target are the research focus of feature extraction. In an Omni-directional vision system built on fish-eye lens, because there is a serious distortion due to the fish-eye lens itself, it's difficult to find good invariant features. A large number of research and experiments show that color-based features have little change in a fish-eye image, but the remaining features such as widely used shape and corners, have obvious change due to the fish-eye lens distortion, and are not suitable as the target characteristics.

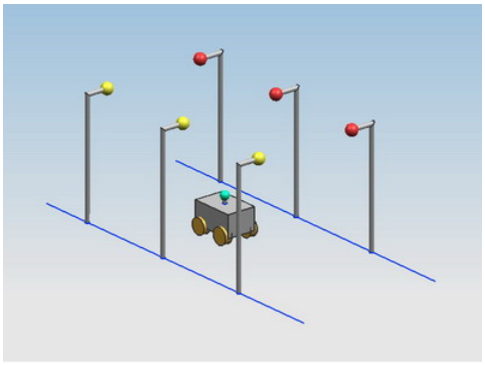


Fig. 5. Two-color sequential landmarks

In theory, considering the large filed of view of the fish-eye lens, using a set of two-color landmark can realize the precise position and navigation. But in fact, when the vehicle goes far away from the landmarks, the landmarks will be located in the image edge of the fish-eye lens, where there is a serious distortion. As a result, not only the landmarks become very small, but also the isometric projection imaging model is no longer applicable, which make the landmarks cannot be used to locate the AGV at this time. Therefore, we arrange equidistant two-color landmarks sequentially along the AGV path as Fig. 5 shows, when a group of landmarks doesn't meet the requirement of imaging model, it will automatically

switch to the next group of landmarks in order to achieve continuous landmark recognition and tracking for the next localization and navigation of the AGV.
In a word, this landmark mode has the topological features close to the natural scenery in indoor and outdoor environment, is simple and practical, easy to layout and maintain, which can effectively improve the efficiency of automatic recognition and tracking in a large scene image with distortion of the fish-eye lens.

3. Fish-eye lens distortion model

Ordinary optical imaging system follows the conventional guideline——similarity, that is, object point is always similar to image point. Optical design tries to ensure this similarity, so does distortion rectification. As a result, come the following imaging formula: (Doucet A, &Godsil S, &Andrieu C. 2000)
When the object is near: $y_0' = \beta y$
When the object is infinitely far: $y_0' = f \tan \omega$
Where y is the height of the object, y_0' is the height of its image, β is lateral magnification, f is the focus of the optical system, ω is half-field angle. But when $\tan 90^\circ = \infty$, similarity can not be met.
In order to cover a wider field angle with a single visual sensor, image compression and deformation is introduced, just as the curves in Fig.6 shows.

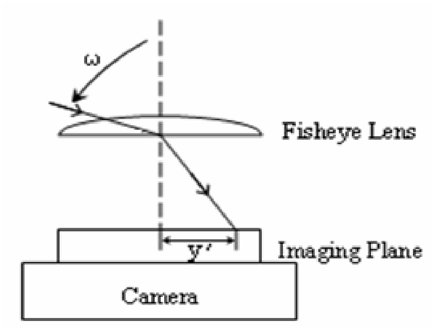


Fig. 6. fish-eye lens distortion

They make the image to achieve deformation to a certain extent, in order to ensure that the solid angle covers expected object space. In addition, the distortion of optical system is determined by the pathway of primary optical axis, so it just causes image deformation, but doesn’t affect image clarity. Despite the obvious distortion, as far as mathematics is considered, there is still one-to-one correspondence between the object and its image, which ensures the correctness and feasibility of the non-similar imaging model. (Fox D. 2003)
Our fisheye lens satisfies isometric projection, the imaging formula is as follows: (Thrun S, &Fox D, &Burgard W. 2001)

$$y' = kf\omega \qquad (0 < k < 1) \tag{1}$$

Where y′ is the height of its image, f is the focus of the optical system, ω is half-field angle.

And $$y' = X^2 + Y^2 \quad \tan\omega = \frac{\sqrt{\left(x_c^2 + y_c^2\right)}}{z_c}$$

$$\begin{cases} u - u_0 = X/d_x \\ v - v_0 = Y/d_y \end{cases} \tag{2}$$

Where (X, Y) is the image coordinate, (xc, yc, zc) is the camera coordinate, (u, v) is the pixel coordinate, and (u0, v0) is the center of the image. According to the equation (2), if (u0, v0) and k are figured out, the conversion relation between the image coordinate and the world coordinate will be fixed, and the rectification of the fisheye image will be accomplished. In a word, the rectification procedure can be divided into two main steps. (Gilks W R, &Berzuini C. 2001)

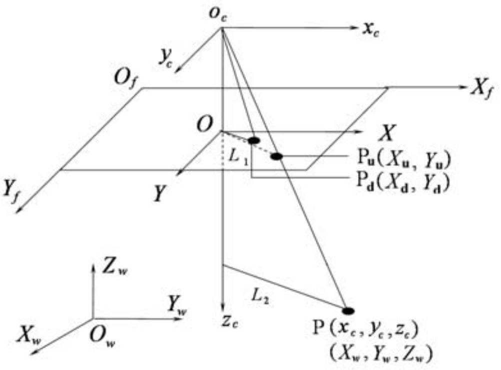


Fig. 7. coordinate system

Firstly, the conversion relation between the camera coordinate (Xc, Oc, Yc) and the world coordinate (Xw, Ow, Yw) should be figured out. As shown in Fig.7, the relation can be expressed as:

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = R\begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + T = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}\begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \quad \text{or}$$

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{3}$$

Where R is the rotation matrix, and T is the translation matrix.

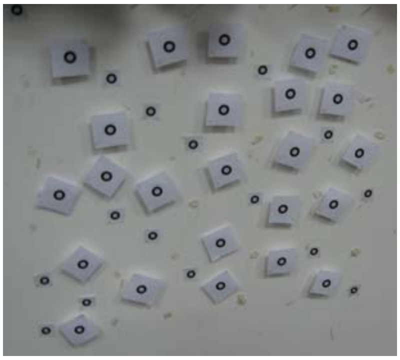


Fig. 8. circular points

The world coordinates of those circular points in Fig.8 are known in advance, if we calculate the image coordinates of those points by extracting their centers, the R and T can be figured out with the equation (3).
Secondly, the coincidence relation between the camera coordinate (Xc, Oc, Yc) and the image coordinate (X, O, Y) should be figured out with the equation (3). (Jung Uk Cho, &Seung Hun Jin. 2006)
Finally, we use 30 circular points to calculate these above equation in the fish-eye distortion rectification, and part of the coordinate data is presented in Table 2. With these equations, the conversion relation between the world coordinate and the image coordinate is figured out, and the center of the image (u0, v0) is (360,288), k = 1.5. Using these parameters and equations, we can implement real-time distortion rectification in DSP.

The world coordinate			The image coordinate	
X	Y	Z	U	V
2026.189	-7127.78	-562.917	298.21	20.87
1756.365	-7141.03	-559.438	578.32	21.32
2174.074	-7118.34	-693.84	143.34	162.08
1900.543	-7133.42	-686.855	433.87	132.65
2028.248	-7124.78	-844.925	281.93	324.59
1759.955	-7138.48	-825.114	601.8	293.74
1970.002	-7121.13	-773.602	348.47	234.49

Table 2. experiment data

4. Improved particle filter

To localize the AGV with landmarks, the coordinate of the beacons in an image should be figured out firstly. So an improved particle filter is employed to track these beacons and get their coordinates.

As an algorithm framework, a particle filter can be used to track multiple objects in the case of nonlinear and non-Gaussian problem. And the object is quite flexible, like man-made or natural landmarks. Particle filter is a Monte Carlo sampling approach to Bayesian filtering. The main idea of the particle filter is that the posterior density is approximated by a set of discrete samples with associated weights. These discrete samples are called particles which describe possible instantiations of the state of the system. As a consequence, the distribution over the location of the tracking object is represented by the multiple discrete particles. (Kwok C, &Fox D, &Meil M. 2004)
In the Bayes filtering, the posterior distribution is iteratively updated over the current state Xt, given all observations Zt = {Z1,..,Zt} up to and including time t, as follows:

$$p(x_t \mid Z_t) = kp(Z_t \mid X_t) \bullet \int_{X_{t-1}} p(X_t \mid X_{t-1}) p(X_{t-1} \mid Z_{t-1}) dx_{t-1} \tag{4}$$

Where p(Zt|Xt) expresses the observation model which specifies the likelihood of an object being in a specific state and p(Xt|Xt-1) is the transition model which specifies how objects move between frames. In a particle filter, prior distribution p(Xt-1|Zt-1) is approximated recursively as a set of N weighted samples , which is the weight for particle . Based on the Monte Carlo approximation of the integral, we can get:

$$p(X_t \mid Z_t) \approx kp(Z_t \mid X_t) \sum_{i=1}^{N} w_{t-1}^{(i)} p(X_t \mid X_{t-1}^{(i)}) \tag{5}$$

Particle filter provides robust tracking of moving objects in a cluttered environment. However, these objects have to be specified manually, and then the particle filter can track them, which is unacceptable for autonomous navigation. Thus, we combine an object recognition algorithm with particle filter, and use the object recognition algorithm to specify landmarks automatically. And once particle filter fails to track the landmarks occasionally, the object recognition algorithm will function to relocate the landmarks. In order to facilitate object recognition, the landmarks are painted certain colors. Based on color histogram, we can find out these landmarks easily.

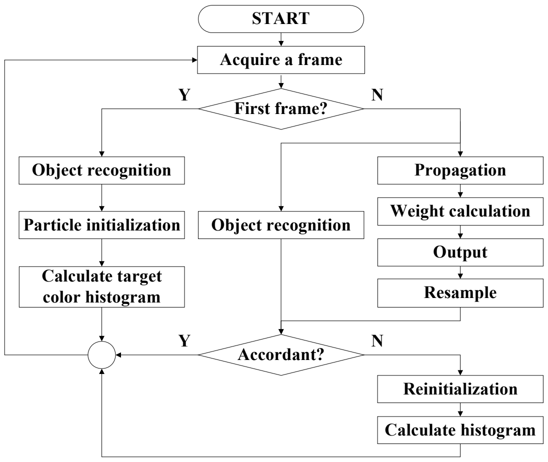


Fig. 9. flow chart of improved Particle Filter

The flow chat of improved PF is shown in Fig.9. When started, a frame is acquired and if it's the first frame, the initialization will be operated. Firstly, the target is recognized by searching in the whole image area based on color histogram, and its location is figured out for next step. Then, the initial particles come out randomly around the above location, and the target color histogram is calculated and saved. Thus, the initialization is finished.
At the next frame, a classical particle filter is carried out. The principal steps in the particle filter algorithm include:

1) Initialization

Draw a set of particles for the prior $p(X_0)$ to obtain $\left\{X_0^{(i)}, w_0^{(i)}\right\}_{i=1}^{N}$.

2) Propagation and Weight calculation
   a) For i=1,…,N, sample $X_k^{(i)}$ from the probability distribution $p\left(X_k^{(i)} \mid X_{k-1}^{(i)}\right)$.
   b) Evaluate the weights $w_k^{(i)} = p\left(Z_k \mid X_k^{(i)}\right)$, i=1,…,N
   c) Normalize the weights $w_k^{(i)} = \dfrac{w_k^{(i)}}{\sum_{j=1}^{N} w_k^{(j)}}$, i=1,…,N

3) Output

Output a set of particles $\left\{X_k^{(i)}, w_k^{(i)}\right\}_{i=1}^{N}$ that can be used to approximate the posterior distribution as $p\left(X_k \mid Z^k\right) \approx \sum_{i=1}^{N} w_k^{(i)} \delta\left(X_k - X_k^{(i)}\right)$ and the estimate as $E_{p\left(g \mid Z^k\right)}\left(f_k\left(X_k\right)\right) \approx \sum_{i=1}^{N} w_k^{(i)} f_k\left(X_k^{(i)}\right)$, where $\delta(g)$ is the Dirac delta function.

4) Resample

Resample particles $X_k^{(i)}$ with probability $w_t^{(i)}$ to obtain N independent and identically distributed random particles $X_k^{(j)}$, approximately distributed according to $P\left(X_k \mid Z^k\right)$.

5) K=k+1, go to step2.

Meanwhile, object recognition is executed and its result is compared with the particle filter. If the recognition result is not accordant with the particle filter, which means that the particle filter fails, the PF will be reinitialized based on the recognition result. According to the combination of object recognition and particle filter, the improved particle filter can keep tracking the target even though the target is blocked or disappeared for a while.

5. Localization and navigation

5.1 Localization

With the coordinates of the landmarks provided by particle filter, we can locate and navigate the AGV automatically.

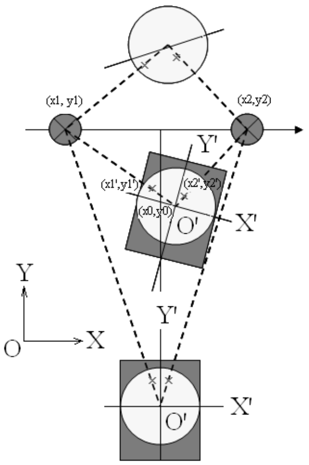


Fig. 10. coordinate conversion

As Fig.10 shows, the world coordinates of the two beacons are (x1, y1) and (x2, y2), and their relative camera coordinates are (x′1, y′1) and (x′2, y′2). In order to steer the AGV, we need to figure out the position of the AGV, (x0, y0) and the angle between the world coordinate and the camera coordinate, θ. According to the coordinate conversion principle, the following equations exist:

$$\begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} = \begin{bmatrix} -\cos\theta & \sin\theta & x_1 \\ -\sin\theta & -\cos\theta & y_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x'_1 \\ y'_1 \\ 1 \end{bmatrix}$$

$$\theta = arctg\frac{y_2 - y_1}{x_2 - x_1} - arctg\frac{y'_2 - y'_1}{x'_2 - x'_1}$$

$$\begin{cases} x_0 = x_1 - kx''_1\cos\theta + ky''_1\sin\theta \\ y_0 = y_1 - kx''_1\sin\theta - ky''_1\cos\theta \\ \theta = arctg\dfrac{y_2 - y_1}{x_2 - x_1} - arctg\dfrac{y''_2 - y''_1}{x''_2 - x''_1} \end{cases} \tag{6}$$

Apparently, we can get (x0, y0) and θ with the above equations and realize real-time location in DSP. Then the position and angle information of the AGV is sent to the built-in computer for navigation.

5.2 Navigation

An incremental PID algorithm is employed as navigation algorithm, which can be expressed as the following equation:

$$\begin{aligned}\Delta u(k) &= u(k) - u(k-1) \\ &= (K_p + K_i + K_d)e(k) - (K_p + 2K_d)e(k-1) + K_d e(k-2)\end{aligned} \tag{7}$$

Suppose $A = (K_p + K_i + K_d), B = -(K_p + 2K_d), C = K_d$, we can get

$$\Delta u(k) == Ae(k) + Be(k-1) + Ce(k-2) \tag{8}$$

Equation (8) shows that we can get each speed increment with *e(k),e(k-1),e(k-2)* in the calculation of *u(k)*, as shown in equation (9).

$$\begin{aligned}&SpeedL = Speed - \Delta u(k) \\ &SpeedR = Speed + \Delta u(k) \\ &e(k) = K_1\theta(k) + K_2 r(k)\end{aligned} \tag{9}$$

Where *SpeedL* and *SpeedR* are the initial velocities of the left and right wheels respectively, the velocity is 0.6m/s. And *K1* and *K2* are the weight of angle deviation and position deviation respectively. We can use equation (9) to figure out the velocities of both wheels, and realize the AGV navigation finally.

6. Navigation experiments

6.1 Experimental conditions

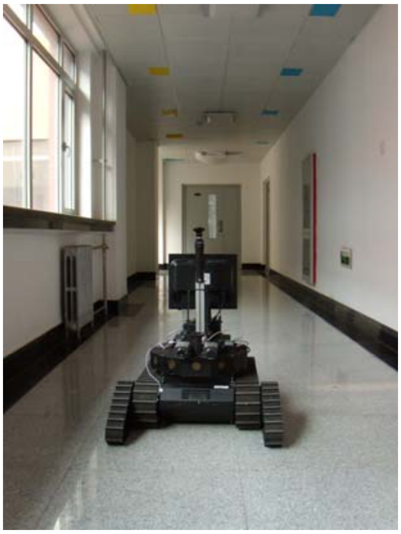


Fig. 11. experiment environment

The experimental environment of the Omni-directional vision sensor for AGV navigation is shown in Fig. 11. The tracked robot is placed in the corridor of a teaching building in this experiment, and the landmarks are fixed on the top of the corridor. Apparently, there are windows, radiator, fire hydrants and other various interferences around the AGV path, so that we can test the robustness of the Omni-directional vision sensor for AGV navigation.

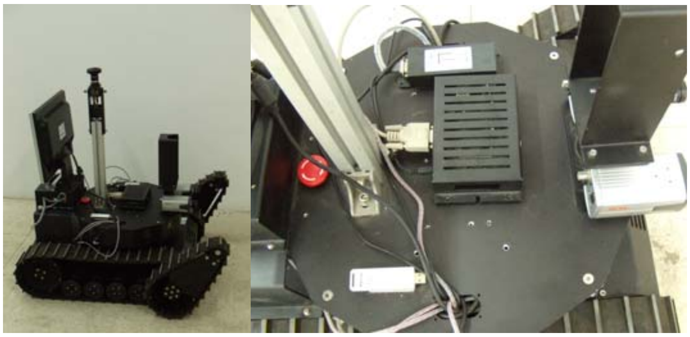


Fig. 12. fish-eye lens camera and embedded image processor

As Fig.12 shows, the fish-eye lens camera is fixed on top of the AGV with a steel frame to ensure that the lens be placed vertically upward. The fish-eye lens is 0.88m above the ground, and it will acquire Omni-directional image of the entire hemisphere domain above the AGV. While the embedded image processor is mounted in a black box, and the box is fixed to the roof of the AGV. Moreover, the black box sets aside hatches for a variety of processor interfaces, and facilitates thermal dispersal of the circuit board with the grid structure on the top of the box.

6.2 Experimental principle
Firstly, the Omni-directional vision sensor based on fish-eye lens outputs standard analog video signal in PAL format, with image resolution of 720 x 576 and frame rate of 25 fps. Secondly, the captured image signal is transmitted to the embedded image processor through a standard video interface, and the DSP on board runs composite particle filter tracking algorithm to achieve real-time recognition and tracking of the two-color sequential landmarks. Thirdly, we use the result of landmark tracking to calculate the position deviation and angle deviation of the AGV path relative to the target path, according to the localization algorithm based on two-color landmarks. Finally, the localization result is sent to the built-in IPC of the AGV as the input of PID control strategy with serial port, and we can change both wheels' speed of the AGV to achieve steering control, and realize the autonomous navigation of the AGV ultimately.

6.3 Experimental result
In accordance with the above experimental principle, we obtained two AGV navigation paths as shown in Fig.13. The blue curve is the target path pre-generated with the two-color sequential landmarks, and the red curve is the AGV navigation path employing angle

deviation as PID control input, while the yellow curve is the AGV navigation path employing both angle deviation and position deviation as PID control input.

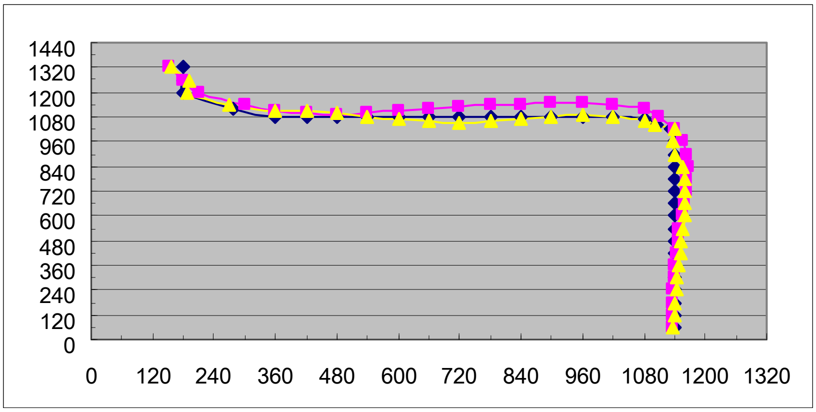


Fig. 13. experimental result (units: cm)

7. Analysis and Conclusion

The autonomous navigation of the AGV can be realized under the control of both angel deviation and position deviation. Experiment shows that the Omni-directional vision sensor can achieve the autonomous navigation of the AGV at a speed of 0.6m/s, and the absolute tracking accuracy is about 10 cm.

In conclusion, Dynamic localization employs a beacon tracker to follow the landmarks in real time during the arbitrary movement of the vehicle. The coordinate transformation is devised for path programming based on time sequence images analysis. The beacon recognition and tracking a key procedure for an omni-vision guided mobile unit. The conventional image processing such as shape decomposition, description, matching, and other usually employed techniques are not directly applicable in omni-vision. PF has been shown to be successful for several nonlinear estimation problems. A beacon tracker based on Particle Filter which offers a probabilistic framework for dynamic state estimation in visual tracking has been developed. In a word, the Omni-directional vision sensor implements the autonomous navigation of the AGV in the structured indoor and outdoor environment.

8. References

Belviken E, &Acklam PJ. (2001). Monte Carlo filters for non-linear state estimation. Automation, 2001, 37(1): pp.177-183.

Crisan D, &Doucet A. (2002). A survey of convergence results on particle filtering methods for practitioners. IEEE Transactions on Signal Processing, 2002, 50(2):pp.736-746.

Doucet A, &Godsil S, &Andrieu C. (2000). On sequential Monte Carlo sampling methods for Bayesian filtering. Statistics and Computing, 2000, 10(3):pp.197-208.

Fox D. (2003). Adapting the sample size in particle filters through KLD sampling. The International Journal of Robotics Research, 2003, 22(12):pp 3-5.

Gilks W R, &Berzuini C. (2001). Following a moving target Monte Carlo inference for dynamic Bayesian models. Journal of the Royal Statistical Society, Series B, 2001, 61(1):pp.127-146.

Jung Uk Cho, &Seung Hun Jin. (2006). A real-time object tracking system using a particle filter. international conference on intelligent robots and systems October 9-15, 2006, Beijing, China.

Kwok C, &Fox D, &Meil M. (2004). A real-time particle filters. Proceedings of the IEEE, 2004, 92(3):pp.469-484.

Thrun S, &Fox D, &Burgard W. (2001). Robust Monte Carlo localization for mobile robots. Artificial Intelligence, 2001, 128(1-2):pp.99-141.

Visual Odometry and mapping for underwater Autonomous Vehicles

Silvia Botelho, Gabriel Oliveira, Paulo Drews,
Mônica Figueiredo and Celina Haffele
Universidade Federal do Rio Grande
Brazil

1. Introduction

In mobile robot navigation, classical odometry is the process of determining the position and orientation of a vehicle by measuring the wheel rotations through devices such as rotary encoders. While useful for many wheeled or tracked vehicles, traditional odometry techniques cannot be applied to robots with non-standard locomotion methods. In addition, odometry universally suffers from precision problems, since wheels tends to slip and slide on the floor, and the error increases even more when the vehicle runs on non-smooth surfaces. As the errors accumulate over time, the odometry readings become increasingly unreliable.
Visual odometry is the process of determining equivalent odometry information using only camera images. Compared to traditional odometry techniques, visual odometry is not restricted to a particular locomotion method, and can be utilized on any robot with a sufficiently high quality camera.
Autonomous Underwater Vehicles(AUVs) are mobile robots that can be applied to many tasks of difficult human exploration[Fleischer00]. In underwater visual inspection, the vehicles can be equipped with down-looking cameras, usually attached to the robot structure [Garcia05]. These cameras capture images from the deep of the ocean. In these images, natural landmarks, also called keypoints in this work, can be detected allowing the AUV visual odometry.
In this text we propose a new approach to AUV localization and mapping. Our approach extract and map keypoints between consecutive images in underwater environment, building online keypoint maps. This maps can be used to robot localization and navigation.
We use Scale Invariant Feature Transform (SIFT), which is a robust invariant method to keypoints detection[David Lowe]. Furthermore, these keypoints are used as landmarks in an online topological mapping. We propose the use of self-organizing maps(SOM) based on Kohonen maps[Teuvo Kohonen] and Growing Cell Structures(GCS)[Bernd Fritzke] that allow a consistent map construction even in presence of noisy information.
First the chapter presents related works on self-localization and mapping. Section III presents a detailed view of our approach with SIFT algorithm and Self-Organizing maps,

followed by implementation, test analysis and results with different undersea features. Finally, the conclusion of the study and future perspectives are presented.

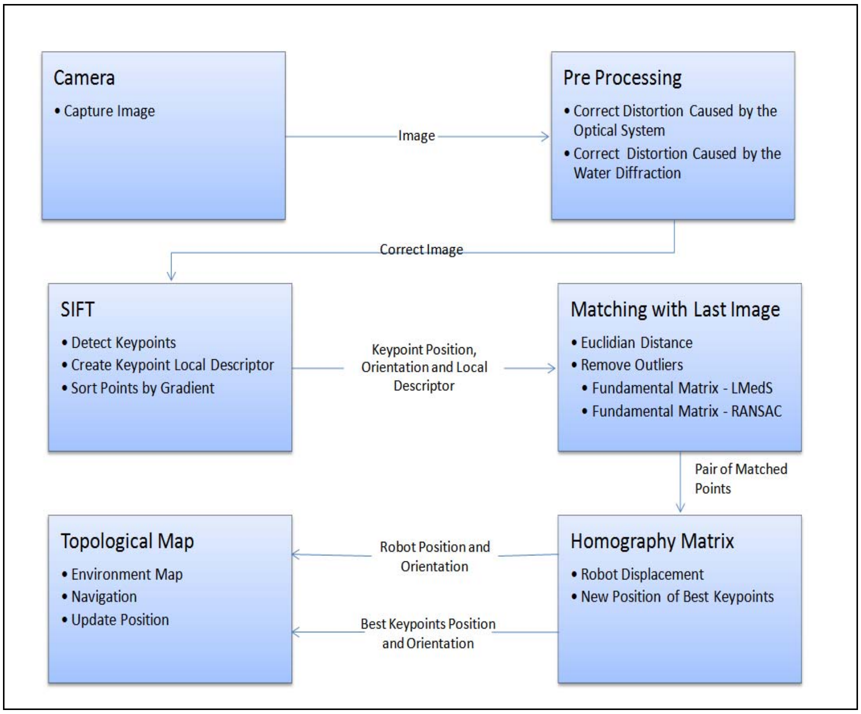
2. Related Works

Localization, navigation and mapping using vision-based algorithms use visual landmarks to create visual maps of the environment. In the other hand the identification of landmarks underwater is complex task due to the highly dynamic light conditions, decreasing visibility with depth and turbidity, and image artifacts like aquatic snow. The extent to which the robot navigates, the map grows in size and complexity, increasing the computational cost and difficult to process in real time. Moreover, the efficiency of the data association, an important stage of the system, decreases as the complexity of the map augment. It is therefore important for these systems, extract a few, but representative, features/keypoints(points of interest) of the environment.
The development of a variety of keypoint detectors was a result of trying to solve the problem of extracting points of interest in image sequences, Shi e Tomasi[Shi94], SIFT[Lowe04], Speed up robust features Descriptor(SURF)[Bay06], affine covariant etc. These proposals have mainly the same approach: extraction of points which represents regions with high intensity gradient. A region represented by then are highly discriminatory and robust to noise and changes in illumination, point of view of the camera, etc.
Some approaches using SIFT for visual indoor Simultaneous Localization and Mapping(SLAM) were made by Se and Lowe [Se02][Se05]. They use SIFT in a stereo visual System to detect the visual landmarks, together with odometry, using ego-motion estimation and Kalman Filter. The Tests were made in structured environments with knew maps.
Several AUVs localization and mapping methods are based on mosaics [Garcia01][Gracias00]. [Mahon04] propose a visual system for SLAM in underwater environments, using the Lucas-Kanade optical filter and extended Kalman filter(EKF), with aid of a sonar. [Nicosevici07] proposes an identification of suitable interest points using geometric and photometric cues in motion video for 3D environmental modelling.
[Booij07] has the most similar approach to the presented in this work. They do visual odometry with classical topological maps based on appearance. In this case, the SIFT method is used in omnidimentional images.

3. A System For Visual Odometry

Figure 1 shows an overview of the approach proposed here. First, the underwater image is captured and pre-processed to removal of radial distortion and others distortions caused by water diffraction. With the corrected image, keypoints are detected and local descriptors for each one these points are computed by SIFT. Each keypoint has a n dimensional local descriptors and global pose informations. A matching stage provides a set of correlated keypoints between consecutive images. Considering all correlated points found, outliers are removed, using RANSAC [Fischler81] and LMedS [Rousseeuw84] algorithms.
The relative motion between frames is estimated, using the correlated points and the homography matrix.

In addition, the keypoints are used to create and train the topological maps. A Growing Cell Structures algorithm is used to create the nodes and edges of the SOM. Each node has a n-dimensional weight. After a training stage, the system provides a topological map, where its nodes represent the main keypoints of the environment.
During the navigation, when a new image is captures, the system calculates its local descriptors, correlating then with the nodes of the current trained SOM. To estimate the pose of the robot (center of the image), we use the correlated points/nodes and homography matrix concept. Thus, it is obtained the global position and orientation of the center of the image, providing the localization of the robot.


Camera
• Capture Image
Image
Pre Processing
• Correct Distortion Caused by the Optical System
• Correct Distortion Caused by the Water Diffraction
Correct Image
SIFT
• Detect Keypoints
• Create Keypoint Local Descriptor
• Sort Points by Gradient
Keypoint Position, Orientation and Local Descriptor
Matching with Last Image
• Euclidian Distance
• Remove Outliers
• Fundamental Matrix - LMedS
• Fundamental Matrix - RANSAC
Pair of Matched Points
Topological Map
• Environment Map
• Navigation
• Update Position
Robot Position and Orientation
Best Keypoints Position and Orientation
Homography Matrix
• Robot Displacement
• New Position of Best Keypoints
Fig. 1. Overview of the system proposed.

3.1 Pre processing

The distortion caused by cameras lenses can be represented by a radial and tangential approximation. As the radial component causes higher distortion, most of the works developed so far corrects only this component [Gracias02].
In underwater environment, there is an additional distortion caused by water diffraction. Equation 1 shows one method to solve this problem [Xu97], where m is the point without radial distortion with coordinates (m_x,m_y), and m_0 the new point without additional distortion; u_0 and v_0 are the central point coordinates. Also, $R=sqrt(m_x^2 + m_y^2)$ and R_0 are defined by 2 with focal distance f.

$$\mathbf{M_0x=m_x+ (R_0/R)(m_x-u_0)} \quad (1)$$

$$\mathbf{M_{0y}=m_y+ (R_0/R)(m_y-v_0)}$$
$$\mathbf{R_0=f\ tan(sin^{-1}(1.33*sin(tan^{-1} R/f)))} \quad (2)$$

3.2 SIFT

The Scale Invariant Feature Transform (SIFT) is an efficient filter to extract and describe keypoints of images [Lowe04]. It generates dense sets of image features, allowing matching under a wide range of image transformations (i.e. rotation, scale, perspective) an important saspect when imaging complex scenes at close range as in the case of underwater vision. The image descriptors are highly discriminative providing bases for data association in several tasks like visual odometry, loop closing, SLAM, etc.

First, the SIFT algorithm uses the Difference-of-Gaussian filter to detect potencial interesting points in a space invariant to scale and rotation. The SIFT algorithm generates s scale space $L(x,y,k\sigma)$ by convolving repeatedly an input image $I(x,y)$ using a variable-scale Gaussian, $G(x,y,\sigma)$, see eq. 3:

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y) \quad (3)$$

SIFT analyzes the images at different scales and extracts the keypoints, detecting scale-invariable image locations. The keypoints represent scale-space extrema in the difference-of-gaussian function $D(x,y,\sigma)$ convolved with the image, see eq. 4:

$$D(x,y,\sigma)=(G(x,y,k\sigma)-G(x,y, \sigma))*I(x,y) \quad (4)$$

Where k is a constant multiplicative factor.

After the keypoints extraction, each feature is associated with a scale and an orientation vector. This vector represents the major direction of the local image gradient at the scale where the keypoint was extracted. The keypoint descriptor is obtained after rotating the nearby area of the feature according to the assigned orientation, thus achieving invariance of the descriptor to rotation. The algorithm analyses images gradients in 4x4 windows around each keypoint, providing a 128 elements vector. This vector represents each set of feature descriptors. For each window a local orientation histogram with 8 bins is constructed. Thus, SIFT maps every feature as a point in a 128-dimension descriptor space.

A point to point distance computation between keypoints in the descriptors space provides the matching. To eliminate false matches, it is used an effective method to compare the smallest match distance to the second-best distance [16], where through a threshold it is selected only close matches.

Futhermore, outliers are removed through RANSAC and LMedS, fitting an homography matrix H [1]. In this chapter, this matrix can be fitted by both RANSAC and LMedS methods [Torr97]. Both methods are considered only if the number of matching points is bigger than a predefined threshold t_m.

3.2 Estimating the Homography Matrix and computing the camera pose

We use the homography concept to provide the camera pose. A homography matrix H is obtained from a set of correct matches, transforming homogenous coordinates into non-homogenous. The terms are operated in order to obtain a linear system [Hartley04], considering the keypoints $(x_1,y_1),..(x_n,Y_n)$ in the image I and $(x_1',y_1'),..(x_n',Y_n')$ in the image I′ obtained by SIFT.
The current global pose of the robot can be estimated using eq. 5, where H_{k+1} is the homography matrix between image I_1 in the initial time and image $I_k + 1$ in the time K+1. The matrix H_1 is defined by the identify matrix 3x3 that consider the robot in the beginning position (0,0).

$$H_{k+1} = \prod_{i=1}^{k} H_{i+1} \tag{5}$$

Thus, the SIFT provides a set of scale invariant keypoints, described by a feature vector. A frame has a *m* keypoints, and each keypoint, X_i, has 128 features, $f_1,...,f_{128}$ and the pose and scale (x,y,s):

$$X_i = f_1, f_2, f_{128}, x, y, s, i = 1,..,m \tag{6}$$

These m vectors are used to obtain a topological map, detailed in the next section.

3.3 Topological Maps

In this work, the vectors extracted from SIFT are used to compose a topological map. This map is obtained using a self-organizing mapping (SOM) based on Kohonen Neural Networks [Kohonen01] and the Growing Cell Structures (GCS) method [Fritzke93]. Like most artificial neural networks, SOMs operate in two modes: training and mapping. Training builds the map using input examples. It is a competitive process, also called vector quantization. A low-dimensional(typically two dimensional) map discretizess the input space of the training samples. The map seeks to preserve the topological properties of the input space. A structure of this map consists of components called nodes or neurons. Associated with each node is a weight vector of the same dimension as the input data vectors and a position in the map space. Nodes are connected by edges, resulting in a (2D) grid.

3.3.1 Building the map

Our proposal operates in Scale Invariant Feature Vectors Space, SIFT space, instead of image space, in other words, our space has n=131 values(128 by SIFT's descriptor vector and 3 by feature's pose). A Kohonen map must be created and trained to represent the space of descriptors. When a new input arrives, the topological map determines the feature vector of the reference node that best matches the input vector. To make clear the method we explain the Kohonen algorithm which is splited in three main modules:

1. **Competition process**, consist in finding in our network a minimum Euclidian distance between de input vector X and de weight vector W, fig 2. If we use the index i(x) to identify the neuron which best matches, called winning node, with a input vector X, so applying the condition below we determine i(x):

$$\mathbf{I(x)=argmin||X\text{-}W_j||}$$

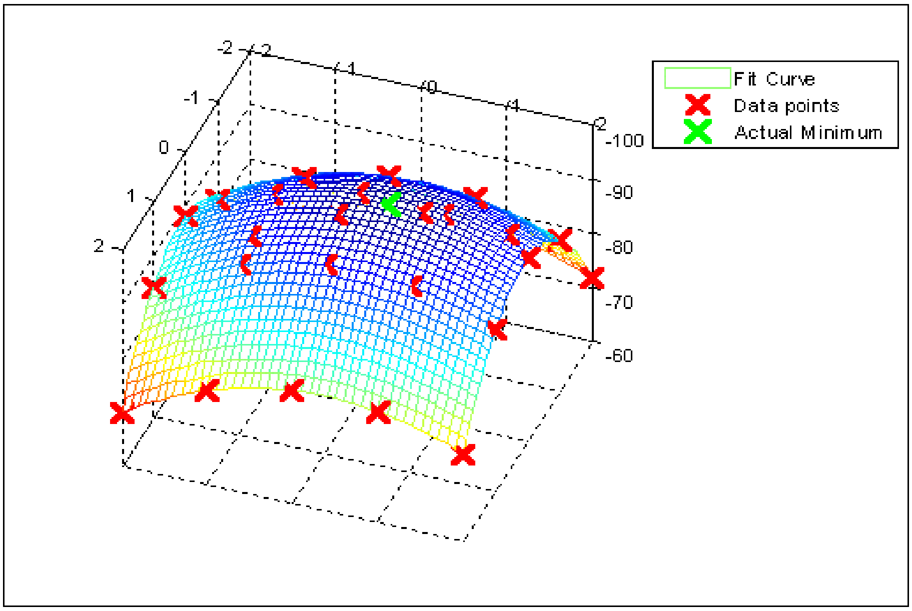


Fig. 2. Minimum distances

2. **Cooperative process,** comprise in a winning node change lateral distance of his neighbors. For that is basically used a Gaussian function, fig 3:

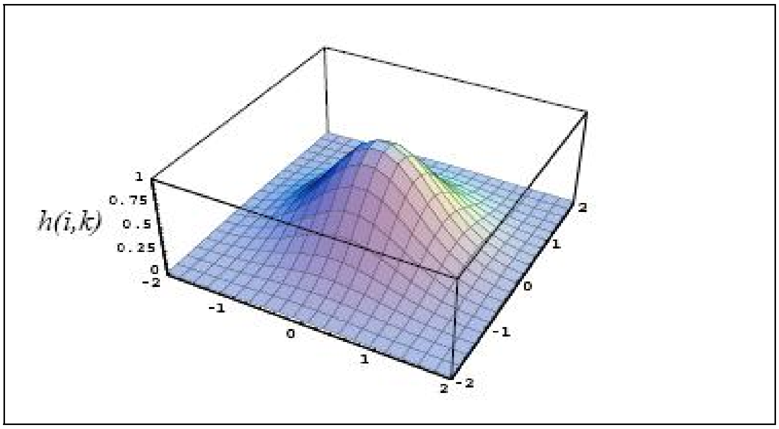


Fig. 3. Gaussian Function

3. **Adaptation process,** this method is responsible for self-organization of a characteristics map. For that it`s necessary a modification of weights vector W in each neuron in relation of a input vector. To perform it is used eq. 7.

$$W_j(n+1)= W_j(n) +\eta(n)h_{j,i(x)}(n)(x- W_j(n)) \tag{7}$$

where η is learning rate and $h_{j,i(x)}$ is Gaussian decay function.

The Growing Cell Structures method allows creation and removal of the nodes during the learning process. The Algorithm constrains the network topology to k-dimensional simplies where by K is some positive integer chosen in advance. In this work, the basic building block and also the initial configuration of each network is a K=2-dimensional simplex. For a given network configuration a number of adaptation steps are used to update the reference vectors of the nodes and to gather local error information at each node. This error information is used to decide where to insert new nodes. A new node is always inserted by splitting the longest edge emanating from the node q with maximum accumulated error. In doing this, additional edges are inserted such that the resulting structure consists exclusively of k-dimensional simplices again, see fig. 4.

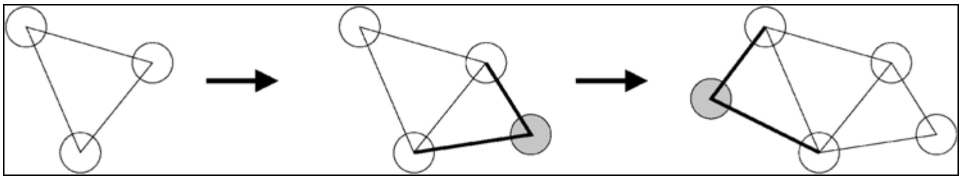


Fig. 4. Illustration of cell insertion

To remove nodes, After a user-specified number of iterations, the cell with the greatest mean Euclidian distance between itself and its neighbors is deleted and any cells within the neighborhood that would be left “dangling” are also deleted, see fig. 5.

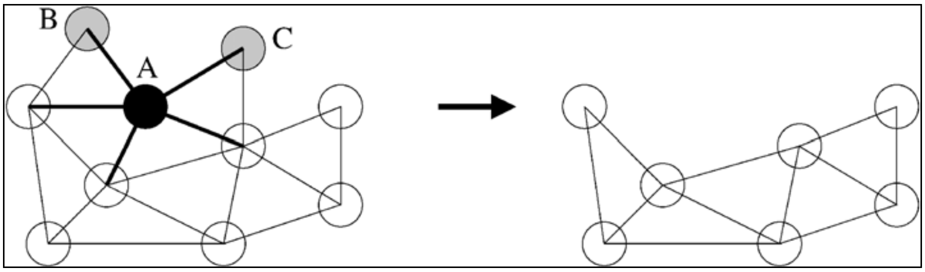


Fig. 5. Illustration of cell deletion: Cell A is deleted. Cells B and C are within the neighborhood of A and would be left dangling by removal of the five connections surrounding A, so B and C are also deleted.

After a set of training steps, the kohonen map represents the descriptors space. This SOM can be used to locate the robot during the navigation.

3.3.2 Location the robot on the map

New frames are captured during the navigation. For each new frame F, SIFT calculates a set of m keypoints X_i, see equation 6. A n=131 dimensional descriptor vector is associated to each keypoint. We use the trained SOM to map/locate the robot in the environment. A mapping stage is runned m times. For each step *i* there will be one single winning neuron, N_i: the neuron whose weight vector lies closest to the input descriptor vector, X_i. This can be simply determined by calculating the Euclidean distance between input vector and weight vectors. After the *m* steps we have a set of *m* winner nodes, N_i, associated with each one feature descriptor, X_i. With the pose information of *m* pairs (X_i, N_i), we can use the homography concept to obtain a linear matrix transformation, H_{SOM}. Equation 8 gives the map localization of center of the frame, $X_C' = (x_c', y_c')$:

$$X_c' = H_{SOM} * X_{C},$$

Where X_C is the position of the center of the frame.
Moreover the final topological map allows the navigation in two ways: through target positions or visual goals. From the current position, graph search algorithms like Dijkstra [Dijkstra59] or A^* algorithm [Dechter85] can be used to search a path to the goal.

4. System Implementation, Tests and Results

In this work, it was developed the robot presented in figure 6. This robot is equipped with a Tritech Typhoon Colour Underwater Video Camera with zoom, a miniking sonar and a set of sensors (altimeters and accelerometers) [Centeno07]. Due this robot is experimental phase, it is impossible to put it to work in the sea. The acquition of some reference to experiements is very hard in this kind of environment, too. Considering this situation, this work use a simulated underwater conditions proposed by [Arrredondo05]. Using it, different undersea features were applied in the images, like turbidity, sea snow, non-linear illumination, and others, simulating different underwater conditions. Table I shows the apllied features (filters).

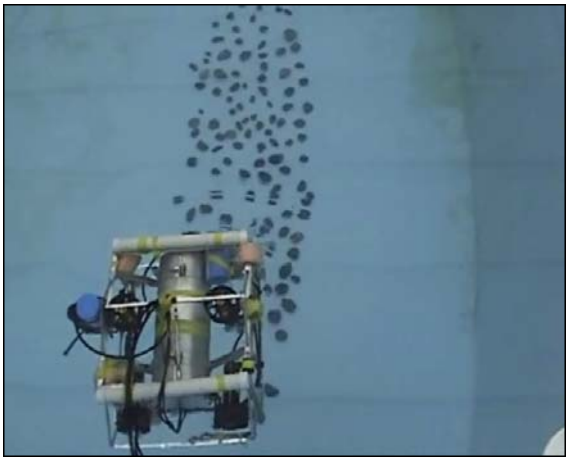


Fig. 6. ROVFURGII in test field

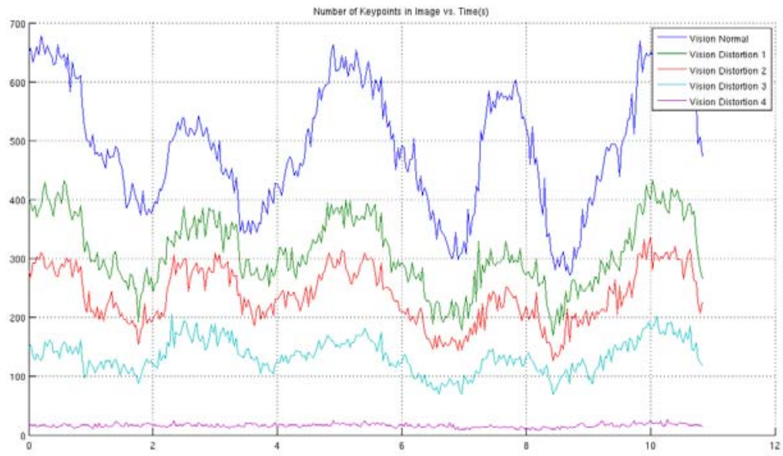
Distortion	1	2	3	4	5
Light Source distance (m)	0.2	0.22	0.25	0.25	0.3
Attenuation value %	0.05	0.05	0.06	0.05	0.05
Gaussian noise σ	2	2	2	4	4
Gray level minimum	20	30	20	20	20
Number of flakes of sea snow	30	30	30	30	30

Table 1. Undersea features for each distortion using in the tests.

The visual system was tested in a desktop Intel Core 2 Quad Q6600 computer with 2 Gb of DDR2-667 RAM. The camera is NTSC standard using 320x240 pixels at a maximum rate of 29.97 frames per second.

4.1 The method in different underwater features

The visual system was tested in five different underwater environments, corresponding the image without distortion and first four filters presented in table I(the effects were artificially added to the image). Figure 7 enumerates the detected and matching keypoints obtained in a sequence of visual navigation. Even though the number of keypoints and correlations has diminished with the quality loss because of underwater conditions, is still possible to localize the robot, according figure 8. In this figure, the motion referencial is represented in blue(legended as "Odometry"), executed by a robotic arm composed by an *Harmonic Drive PSA-80* actuator with a couple encoder supplying angular readings in each 0.651 ms, with a camera coupled to this. It allows the reference system a good precision, 50 pulses per revolution. Therewith, it is possible to see that our approach is robust to underwater environment changes. All graphics in this chapter use centimeter as metric unit, including figure 8.



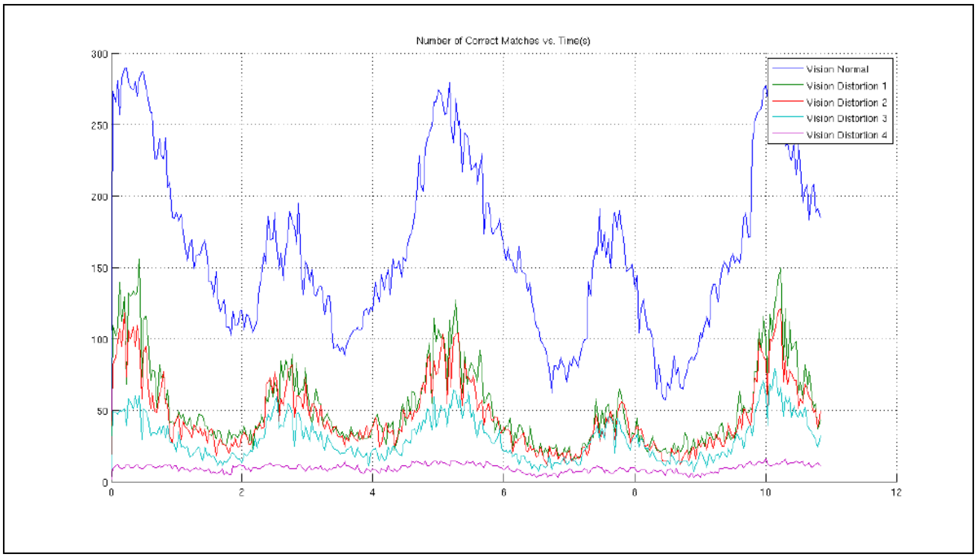
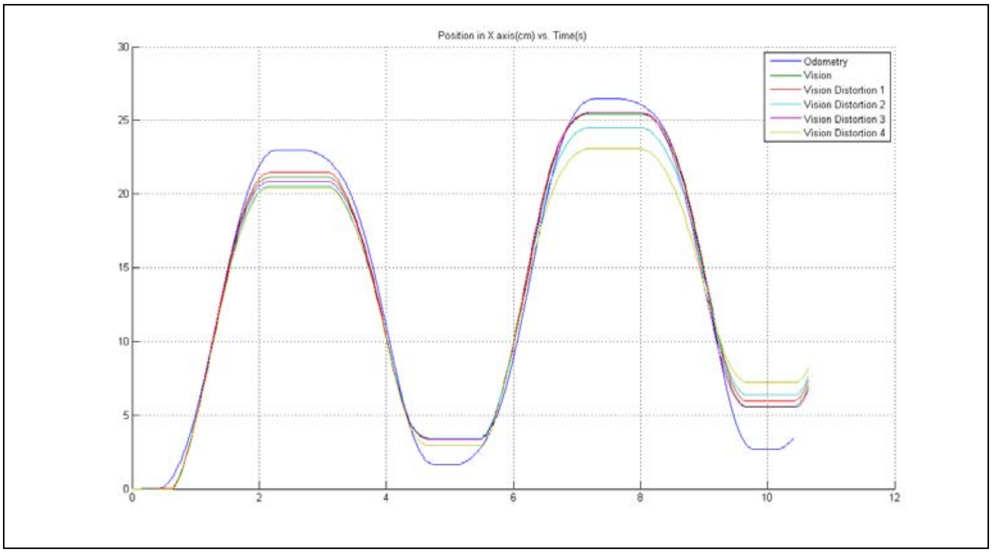


Fig. 7. Number of keypoints detected and true correlation during the robotic arm movement.



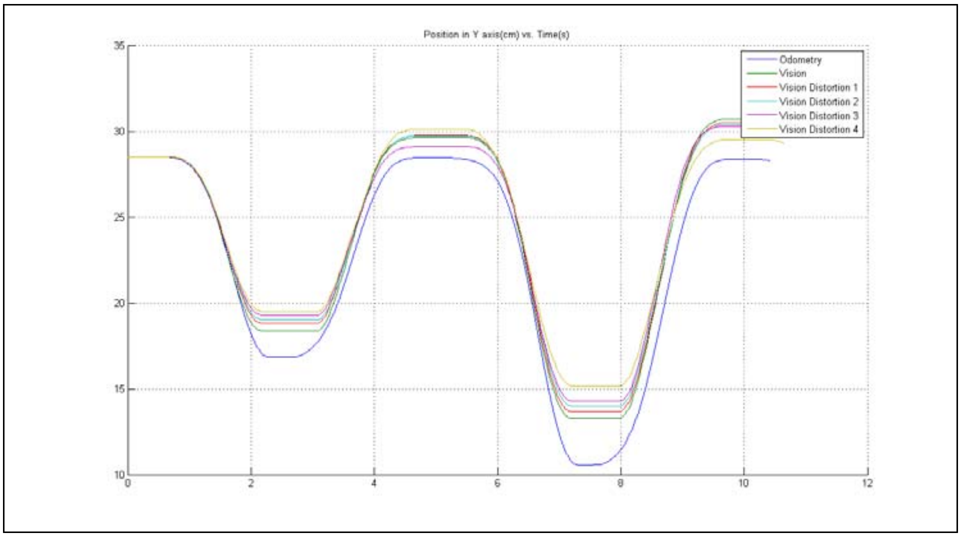


Fig. 8. Position determinated by the robotic arm odometry and a visual system, without and with distortion.

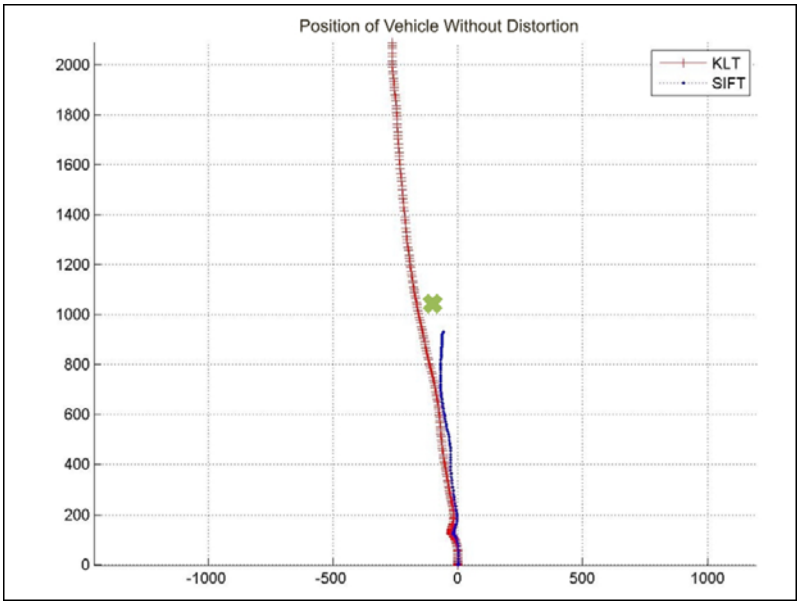
4.2. Online Robotic Localization

Tests were performed to evaluate the SIFT algorithm performance considering a comparison with another algorithm for robotic localization in underwater environment: KLT [Plakas00][Tommasini98][Tomasi91][Shi94].
Figure 9 shows the performance results using and KLT methods. SIFT has obtained an average rate of 4.4 fps over original images, without distortion, and a rate of 10.5 fps with the use of filter 5, the worst distortion applied. KLT presented higher averages, 13.2 fps and 13.08 fps, respectively. Note that SIFT has worst performance in high quality images because the large amount of detected points and, consequently, because the higher number of descriptors to be processed. The KLT, instead, keeps an almost constant performance. However, due to he slow dynamic associated with undersea vehicle motion, both methods can be applied to online AUV SLAM. The green cross represent the real final position and the metric unit is centimeter.
The SIFT results related to the robot localization were considered satisfactory, even with extreme environment distortions (filter 5). In the other hand, KLT gives insatisfying results for both cases, onde it is too much susceptible to the robot`s depth variation, or image scale, that occurs constantly in the AUV motion despite the depth control.

4.3. Robustness to Scale

A set of tests were performed to estimate the robustness of the proposed system to the sudden scale variantion. In this case, a translation motion with height variation was performed with the camera to simulate a deeper movement of the robot in critical conditions.
The figure 10 shows the SIFT results, considered satisfactory, even in critical water conditions. Considering the use of some filters in extreme conditions, SIFT is superior to

KLT although it shows an inexistent movement in Y axis. Over the tests, SIFT has shown an average rate of 6.22 fps over original images captured by the camera and a rate of 7.31 fps using filter 1 and 10.24 fps using filter 5. The KLT have shown 12.5, 10.2 and 11.84 fps, respectively. The green cross represent the real final position, is the same for all graphics in figure 10, the metric is centimeter.



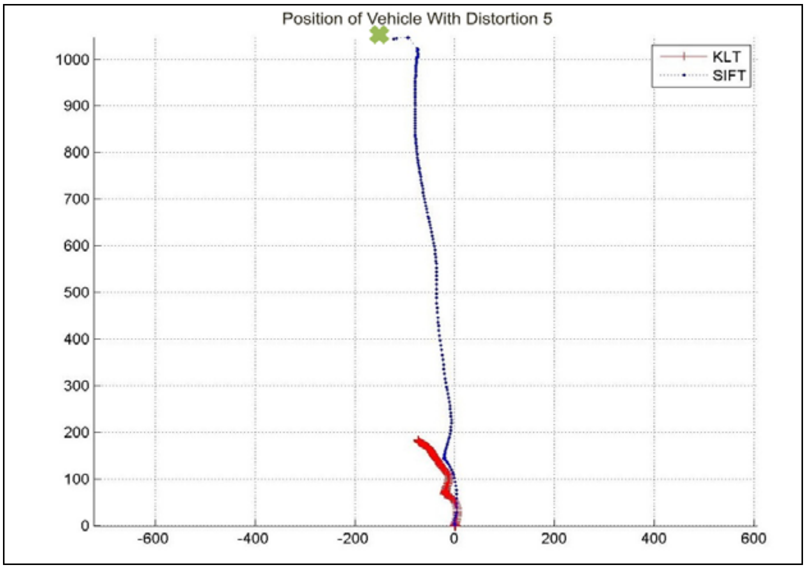
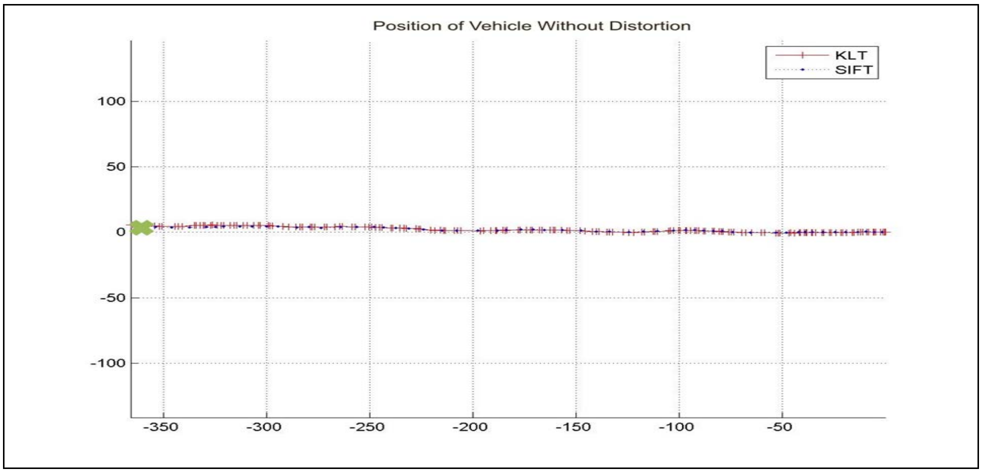
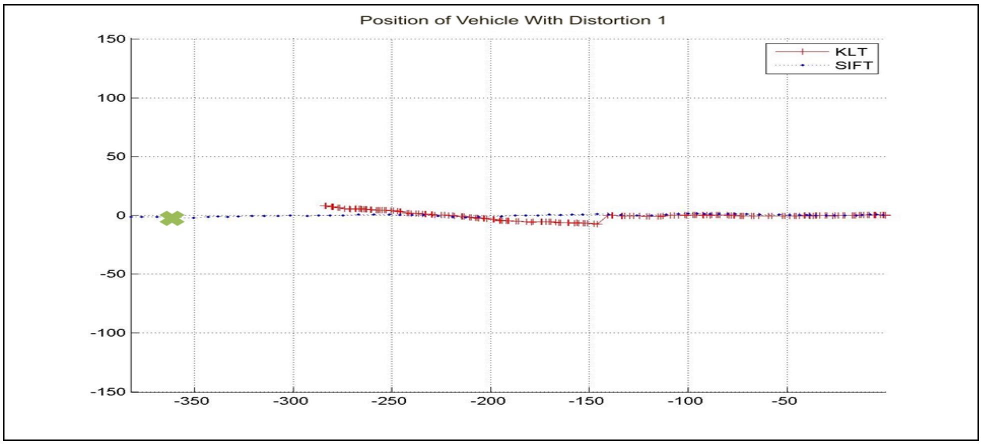


Fig. 9. Real Robot Localization in online system, without and with artificial distortion.





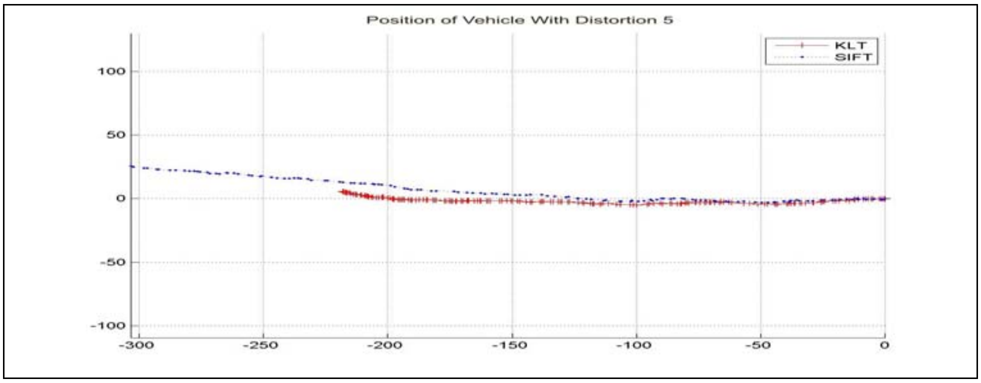


Fig. 10. Localization with translation and scale movement without and with distortion.

4.4. Topological Maps

Tests to validate the mapping system proposed were performed. For example, during a navigation task s set of 1026 frames were captured. From these frames, a total of 40.903 vectors are extracted from SIFT feature algorithm.

To build the map, 1026 frames and 40903 keypoints are presented to the SOM. Figure 11 show the final 2D map, discretizing the input space of the training samples.

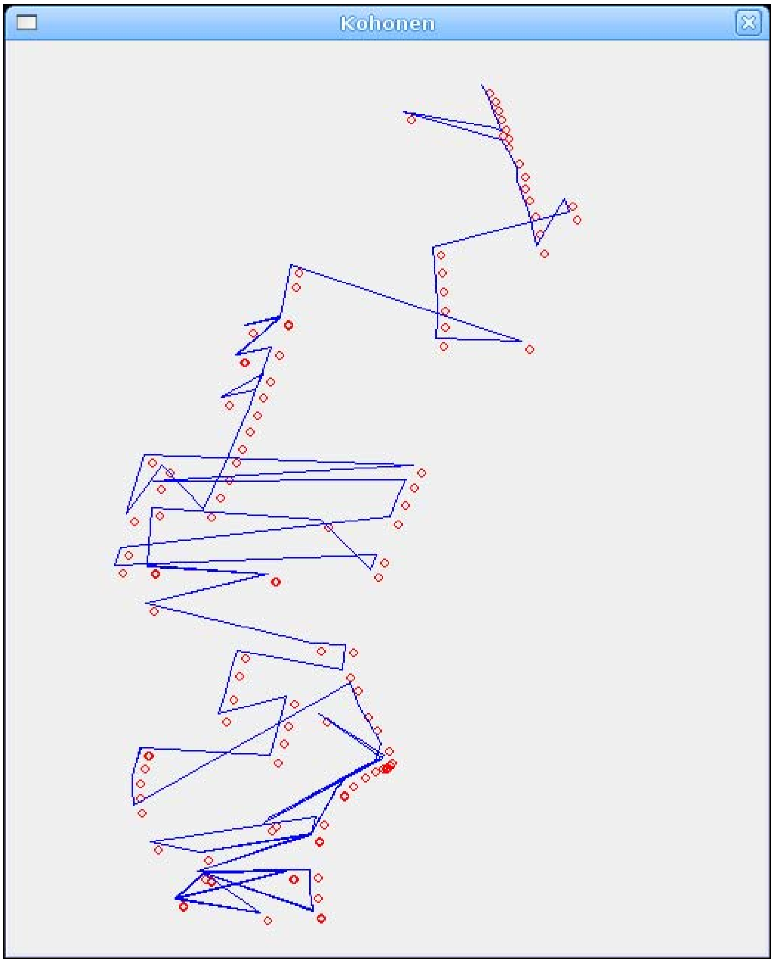


Fig. 11. Topological Map generated by ROVFURGII in movement.

a) Building the map: when a new keypoint arrives, the topological map determines the feature vector of the reference node that best matches the input vector. The Growing Cell Structures (GCS) method allows the creation and removal of the nodes during the learning process. Table II shows intermediate GCS adaptation steps with number of frames, keypoints and SOM nodes. After the training stage

(1026 frames), the kohonen map represents the relevant and noise tolerant descriptors space using a reduced number of nodes. This SOM can be used to locate the robot during the navigation.

Frames	Keypoints	Nodes
324	20353	280
684	35813	345
1026	44903	443

Table 2. building the map with gcs algorithm

b) Location of robot on the map: New frames are captured during the navigation. We use the trained SOM to map/locate the robot in the environment. Figure 12 shows the estimated position of a navigation task. In this task the robot crosses three times the position 0.0. In this figure we can see the position estimated by both the SOM map (blue) and only by visual odometry (red). In the crossings, table III shows the normalized errors of positioning in each of the methods. The reduced error associated with the SOM localization validate the robusteness of topological approach.

Visual Odometry	SOM
0.33	0.09
0.68	0.35
1.00	0.17

Table 3. Normalized localization errors of only visual odometry and som.

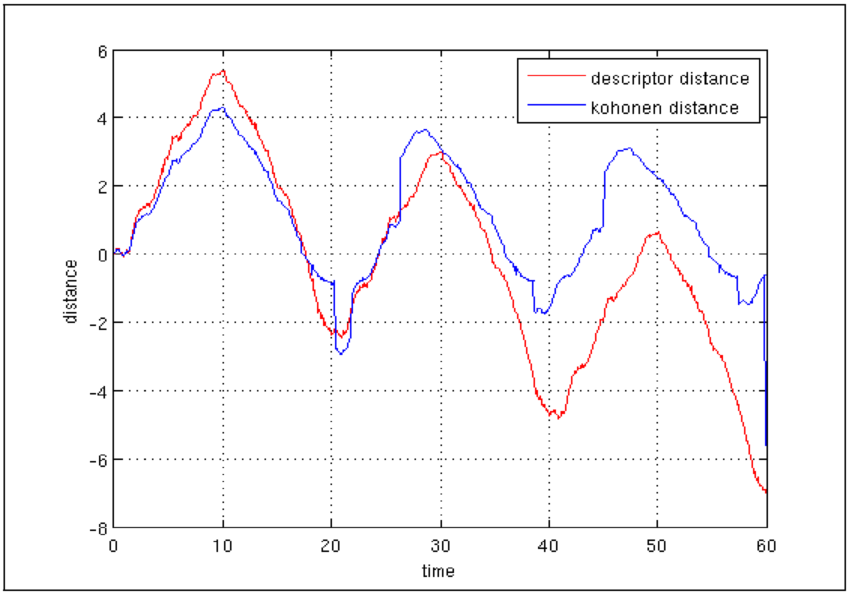


Fig. 12. Distance Y generated by ROVFURGII in movement.

5. Conclusion

This work proposed a new approach to visual odometry and mapping of a underwater robot using only online visual information. This system can be used either in autonomous inspection tasks or in control assistance of robot closed-loop, in case of a human remote operator.
A set of tests were performed under different underwater conditions. The effectiveness of our proposal was evaluated inside a set of real scenario, with different levels of turbidity, snow marine, non-uniform illumination and noise, among others conditions. The results have shown the SIFT advantages in relation to others methods, as KLT, in reason of its invariance to illumination conditions and perspective transformations. The estimated localization is robust, comparing with the vehicle real pose.
Considering time performance, our proposal can be used to online AUV SLAM, even in very extreme sea conditions.
The correlations of interest points provided by SIFT were satisfying, even though with the presence of many *outliers*, i.e., false correlations. The proposal of use of fundamental matrix estimated in robust ways in order to remove *outliers through RANSAC and LMedS algorithms.*
The original iintegration of SIFT and topological maps with GCS for AUV navigation is a promising field. The topological mapping based on Kohonen Nets and GCS showed potencial potential to underwater SLAM applications using visual information due to its robustness to sensory impreciseness and low computational cost. The GCS stabilizes in a limited number of nodes sufficient to represent a large number if descriptors in a long sequence of frames. The SOM localization shows good results, validating its use with visual odometry.

6. References

[Arredondo05] Miguel Arredondo and Katia Lebart. A methodology for the systematic assessment of underwater video processing algorithms. Oceans - Europe, 1:362–367, June 2005.

[Bay06] H. Bay, T. Tuytelaars, and L.booktitle = SURF: Speeded Up Robust Features Van Gool. Surf: Speeded up robust features. In 9th European Conference on Computer Vision, pages 404–417, 2006.

[Booij07] O. Booij, B. Terwijn, Z. Zivkovic, and B. Krose. Navigation using an appearance based topological map. In IEEE International Conference on Robotics and Automation, pages 3927–3932, April 2007.

[Centeno07] Mario Centeno. Rovfurg-ii: Projeto e construção de um veículo subaquático não tripulado de baixo custo. Master's thesis, Engenharia Oceânica - FURG, 2007.

[Dechter85] Rina Dechter and Judea Pearl. Generalized best-first search strategies and the optimality af a*. Journal of the Association for Computing Machinery, 32(3):505–536, July 1985.

[Dijkstra59] Edsger W. Dijkstra. A note on two problems in connexion with graphs. Numerische Mathematik, 1:269–271, 1959.

[Fischler81] Martin Fischler and Robert Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6):381–395, 1981.

[Fleischer00] Stephen D. Fleischer. Bounded-Error Vision-Based Navigation of Autonomous Underwater Vehicles. PhD thesis, Stanford University, 2000.

[Fritzke93] Bernd Fritzke. Growing cell structures - a self-organizing network for unsupervised and supervised learning. Technical report, University of California - Berkeley, International Computer Science Institute, May 1993.

[Garcia01] Rafael Garcia, Xavier Cufi, and Marc Carreras. Estimating the motion of an underwater robot from a monocular image sequence. In IEEE/RSJ International Conference on Intelligent Robots and Systems, volume 3, pages 1682–1687, 2001.

[Garcia05] Rafael Garcia, V. Lla, and F. Charot. Vlsi architecture for an underwater robot vision system. In IEEE Oceans Conference, volume 1, pages 674–679, 2005.

[Gracias02] N. Gracias, S. Van der Zwaan, A. Bernardino, and J. Santos-Vitor. Results on underwater mosaic-based navigation. In IEEE Oceans Conference, volume 3, pages 1588–1594, october 2002.

[Gracias00] Nuno Gracias and Jose Santos-Victor. Underwater video mosaics as visual navigation maps. Computer Vision and Image Understanding, 79(1):66–91, July 2000.

[Hartley04] Richard Hartley and Andrew Zisserman. Multiple View Geometry in Computer Vision. Cambridge University Press, 2004.

[Kohonen01] Teuvo Kohonen. Self-Organizing Maps. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.

[Lowe04] David Lowe. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(2):91–110, 2004.

[Mahon04] I. Mahon and S. Williams. Slam using natural features in an underwater environment. In Control, Automation, Robotics and Vision Conference, volume 3, pages 2076–2081, December 2004.

[Nicosevici07] T Nicosevici, R. García, S. Negahdaripour, M. Kudzinava, and J Ferrer. Identification of suitable interest points using geometric and photometric cues in motion video for efficient 3-d environmental modeling. In International Conference in Robotic and Automation, pages 4969–4974, 2007.

[Plakas00] K. Plakas and E. Trucco. Developing a real-time, robust, video tracker. In MTS/IEEE OCEANS Conference and Exhibition, volume 2, pages 1345–1352, 2000.

[Rousseeuw84] Peter Rousseeuw. Least median of squares regression. Journal of the American Statistics Association, 79(388):871–880, December 1984.

[Se02] Stephen Se, David Lowe, and James Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. The International Journal of Robotics Research, 21(8):735–758, 2002.

[Se05] Stephen Se, David Lowe, and James Little. Vision-based global localization and mapping for mobile robots. IEEE Transactions on Robotics, 21(3):364–375, June 2005.

[Shi94] Jianbo Shi and Carlo Tomasi. Good features to track. In IEEE Conference on Computer Vision and Pattern Recognition, pages 593–600, 1994.

[Tomasi91] Carlos Tomasi and Takeo Kanade. Detection and tracking of point features. Technical report, Carnegie Mellon University, April 1991.

[Tommasini98] T. Tommasini, A. Fusiello, V. Roberto, and E. Trucco. Robust feature tracking in underwater video sequences. In IEEE OCEANS Conference and Exhibition, volume 1, pages 46–50, 1998.

[Torr97] P. H. S. Torr and D. W. Murray. The development and comparison of robust methods for estimating the fundamental matrix. International Journal of Computer Vision, 24(3):271 – 300, 1997.

[Xu97] Xun Xu and Shahriar Negahdaripour. Vision-based motion sensing for underwater navigation and mosaicing of ocean floor images. In MTS/IEEE OCEANS Conference and Exhibition, volume 2, pages 1412–1417, October 1997.

A Daisy-Chaining Visual Servoing Approach with Applications in Tracking, Localization, and Mapping

S. S. Mehta, W. E. Dixon
University of Florida, Gainesville
USA

G. Hu
Kansas State University, Manhattan
USA

N. Gans
University of Texas, Dallas
USA

1. Introduction

Recent advances in image processing, computational technology and control theory are enabling vision-based control, localization and mapping to become more prevalent in autonomous vehicle applications. Instead of relying solely on a global positioning system (GPS) or inertial measurement units (IMU) for navigation, image-based methods are a promising approach to provide autonomous vehicles with position and orientation (i.e., pose) information. Specifically, rather than obtain an inertial measurement of an autonomous vehicle, vision systems can be used to recast the navigation, localization, control and mapping problems in terms of the image space.

Applications involving localization and mapping using camera as a sensor are often described as Visual Simultaneous Localization and Mapping (VSLAM) (Davison et al., 2007; Eustice et al., 2005; Goncalves et al., 2005; Jensfelt et al., 2006; Jung & Lacroix, 2003; Kim & Sukkarieh, 2003; Se et al., 2002), wherein the camera is the main sensor used to estimate the location of a robot in the world, as well as estimate and maintain estimates of surrounding terrain or features. There are many overlapping ways to categorize VSLAM approaches. Some authors (e.g., (Eustice et al., 2005; Jensfelt et al., 2006; Se et al., 2002)) make a distinction between "local VSLAM" and "global VSLAM". Many VSLAM approaches use probabilistic filters (e.g., extended Kalman filter or particle filter) (Davison et al., 2007; Eustice et al., 2005; Jensfelt et al., 2006; Jung & Lacroix, 2003; Kim & Sukkarieh, 2003), typically estimating a state vector composed of the camera/robot position, orientation and velocity, and the 3D coordinates of visual features in the world frame. An option to a filtered based approach is the use of epipolar geometry (Goncalves et al., 2005; Se et al., 2002). A final possible category are methods that

build a true 3D map (i.e., a map that is easily interpreted by a human being such as walls or topography) (Eustice et al., 2005; Jensfelt et al., 2006; Jung & Lacroix, 2003; Kim & Sukkarieh, 2003; Se et al., 2002), and those that build a more abstract map that is designed to allow the camera/robot to accurately navigate and recognize its location, but not designed for human interpretation.
From the navigation perspective, vision-based pose estimation has motivated results such as (Baker & Nayar, 1999; Burschka & Hager, 2001; Chen et al., 2006; Das et al., 2001; Dixon et al., 2001; Fang et al., 2005; Hagar et al., 1998; Kim et al., 2001; Ma et al., 1999; Song & Huang, 2001) and others, where a camera provides feedback information to enable autonomous navigation of a control agent. See (Chen et al., 2006) for a detailed review of these and other related results. Typically these results are focused on the regulation result, and in all the results the targets are static with respect to the moving camera or the camera is stationary and recording images of the moving control agent.
Vision-based cooperative control methods can involve a moving camera supplying regulation/tracking control input to a moving control agent. A practical example application of this scenario is an airborne camera attached to a remote controlled aircraft that is used to determine a desired video of an unmanned ground vehicle (UGV) moving in a terrain, and then another moving camera (which does not have to follow the same trajectory as the previous camera) is used to relate and control the pose of a moving UGV with respect to the recorded video. The challenge here is to account for the relative velocity between the moving camera and the moving UGV. Also, the reference objects (or features) used to evaluate the pose can leave the camera's field-of-view (FOV) while new reference object enters the FOV. In this scenario, the vision-based system should be intelligent to switch from the leaving reference object to a new reference object to provide the pose information to the controller.
This chapter uses a new daisy-chaining method for visual servo tracking control of a rigid-body object, such as an UGV, while providing localization of the moving camera and moving object in the world frame, and mapping the location of static landmarks in the world frame. Hence, this approach can be used in control and local VSLAM of the UGV, with applications toward path planning, real time trajectory generation, obstacle avoidance, multi-vehicle coordination control and task assignment, etc. By using the daisy-chaining strategy, the coordinates of static features out of the FOV can also be estimated. The estimates of static features can be maintained as a map, or can be used as measurements in existing VSLAM methods.
Section 2 introduces the imaging model, geometric model used in this chapter, as well as introduces the daisy-chaining method as applied to the case of controlling a six-DOF planar object through visual data from a moving camera and fixed reference camera. These results are extended to the case of an UGV with nonholonomic constraints and a moving camera and moving reference camera in Section 3. The efforts of previous sections are then brought to bear on a tracking and mapping application, where the UGV is controlled to track a trajectory that takes the vehicle outside of the initial FOV of the camera. The daisy-chaining approach must be extended to allow for new fixed landmarks to enter the FOV and related to previous landmarks and the UGV.

2. Daisy-Chaining Based Tracking Control

In this section, a visual servo tracking controller is developed for a moving six-DOF agent based on daisy-chained image feedback from a moving camera. The objective is to enable a controlled agent to track a desired trajectory determined by a sequence of prerecorded images from a stationary camera. To achieve this result, several technical issues must be resolved including: discriminating the relative velocity between the moving camera and the moving agent, compensating for the unknown time-varying distance measurement from the camera to the agent, relating the unknown attitude of the control agent to some measurable signals, and using the unit quaternion to formulate the rotation motion and rotation error system. The relative velocity issue is resolved by utilizing multi-view image geometry to daisy-chain homography relationships between the moving camera frame and the moving agent coordinate frames. By using the depth ratios obtained from the homography decomposition, the unknown depth information is related to an unknown constant that can be compensated for by a Lyapunov-based adaptive update law. Lyapunov-based methods are provided to prove the adaptive asymptotic tracking result.

2.1 Problem Scenario

Over the past decade, a variety of visual servo controllers have been addressed for both camera-to-hand and camera-in-hand configurations (e.g., see (Allen et al., 1993; Hager et al., 1995; Hutchinson et al., 1996; Wiiesoma et al., 1993)). Typical camera-to-hand and camera-in-hand visual servo controllers have required that either the camera or the target remain stationary so that an absolute velocity can be determined and used in the control development. For the problem of a moving camera tracking a moving target (i.e. control of relative pose/velocity), integral control or predictive Kalman filters have been used to overcome the unknown target velocity (Bensalah & Chaumette, 1995; Papanikolopoulos et al., 1993). In contrast to these methods, the development in this section and our previous preliminary work in (Hu, Mehta, Gans & Dixon, 2007; Mehta, Dixon, MacArthur & Crane, 2006; Mehta, Hu, Gans & Dixon, 2006) is motivated by the problem when the camera and the target are moving. A practical example application of this scenario is an airborne camera attached to a remote controlled aircraft that is used to determine pose measurements of an UGV and then relay the information to the UGV for closed-loop control.

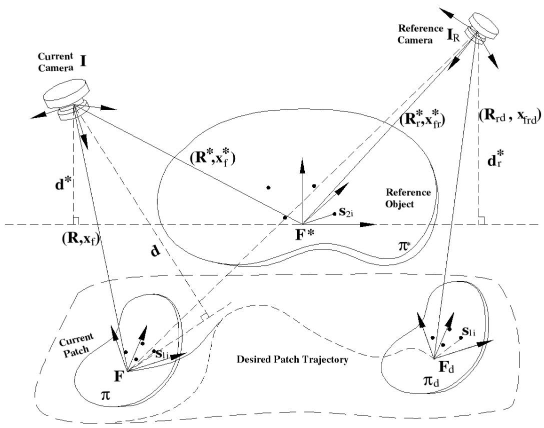


Fig. 1. Geometric model for a moving camera, moving target and stationary reference camera.

The scenario examined in this section is depicted in Fig. 1, where various coordinate frames are defined as a means to develop the subsequent Euclidean reconstruction and control methods. In Fig. 1, a stationary coordinate frame $\mathcal{I}_R$ is attached to a camera and a time-varying

coordinate frame $\mathcal{F}_d$ is attached to some mobile agent (e.g., an aircraft, a ground vehicle, a marine vessel). The agent is identified in an image through a collection of feature points that are assumed (without loss of generality[1]) to be coplanar and non-collinear (i.e., a planar patch of feature points). The camera attached to $\mathcal{I}_R$ a priori records a series of snapshots (i.e., a video) of the motion of the coordinate frame $\mathcal{F}_d$ until $\mathcal{F}_d$ comes to rest. A stationary coordinate frame $\mathcal{F}^*$ is attached to another planar patch of feature points that are assumed to be visible in every frame of the video recorded by the camera. For example, the camera attached to $\mathcal{I}_R$ is on-board a stationary satellite that takes a series of snapshots of the relative motion of $\mathcal{F}_d$ with respect to $\mathcal{F}^*$. Therefore, the desired motion of $\mathcal{F}_d$ can be encoded as a series of relative translations and rotations with respect to the stationary frame $\mathcal{F}^*$ a priori. Spline functions or filter algorithms can then be used to generate a smooth desired feature point trajectory as described in (Chen et al., 2005).
Fig. 1 also depicts a time-varying coordinate frame $\mathcal{I}$ that is attached to another camera (e.g., a camera attached to a remote controlled aircraft), and a time-varying coordinate frame $\mathcal{F}$ that is attached to the current pose of the planar patch. The camera attached to $\mathcal{I}$ captures snapshots of the planar patches associated with $\mathcal{F}$ and $\mathcal{F}^*$, respectively. The a priori motion of $\mathcal{F}_d$ represents the desired trajectory of the coordinate system $\mathcal{F}$, where $\mathcal{F}$ and $\mathcal{F}_d$ are attached to identical objects, but at different points in time. The camera attached to $\mathcal{I}_R$ can be a different camera (with different calibration parameters) as the camera attached to $\mathcal{I}$. Based on these coordinate frame definitions, the problem considered in this section is to develop a kinematic controller for the object attached to $\mathcal{F}$ so that the time-varying rotation and translation of $\mathcal{F}$ converges to the desired time-varying rotation and translation of $\mathcal{F}_d$, where the motion of $\mathcal{F}$ is determined from the time-varying overhead camera attached to $\mathcal{I}$.

2.2 Geometric Relationships

Relationships between the various coordinate frames are summarized in Table I. In Table I, $R(t)$, $R^*(t)$, $R_r(t)$, $R'(t)$, $R_{rd}(t)$, $R_r^* \in SO(3)$ denote rotation matrices, and $x_{fr}(t)$, $x'_{fr}(t)$, $x_{frd}(t)$, $x^*_{fr} \in \mathbb{R}^3$ denote translation vectors. From Fig. 1, the translation $x'_{fr}(t)$ and the rotation $R'(t)$ can be expressed as

$$\begin{aligned} x'_{fr} &= x^*_{fr} + R_r^* R^{*T}(x_f - x_f^*) \\ R' &= R_r^* R^{*T} R. \end{aligned} \tag{1}$$

As illustrated in Fig. 1, π, π_d and π^* denote the planes of feature points associated with $\mathcal{F}$, $\mathcal{F}_d$, and $\mathcal{F}^*$, respectively. The constant Euclidean coordinates of the i-th feature point in $\mathcal{F}$ (and also $\mathcal{F}_d$) are denoted by $s_{1i} \in \mathbb{R}^3$ $\forall i = 1, 2, \cdots, n$ $(n \geq 4)$, and $s_{2i} \in \mathbb{R}^3$ $\forall i = 1, 2, \cdots, n$ denotes the constant Euclidean coordinates of the i-th feature point in $\mathcal{F}^*$. From the geometry between the coordinate frames depicted in Fig. 1, the following relationships

[1] Image processing techniques can often be used to select coplanar and non-collinear feature points within an image. However, if four coplanar target points are not available then the subsequent development can also exploit the virtual parallax method (Boufama & Mohr, 1995; Malis & Chaumette, 2000) where the non-coplanar points are projected onto a virtual plane.

Motion	Frames
$R(t), x_f(t)$	$\mathcal{F}$ to $\mathcal{I}$ in $\mathcal{I}$
$R^*(t), x_f^*(t)$	$\mathcal{F}^*$ to $\mathcal{I}$ in $\mathcal{I}$
$R_r(t), x_{fr}(t)$	$\mathcal{I}$ to $\mathcal{I}_R$
$R^{'}(t), x_{fr}^{'}(t)$	$\mathcal{F}$ to $\mathcal{I}_R$ in $\mathcal{I}_R$
R_r^*, x_{fr}^*	$\mathcal{F}^*$ to $\mathcal{I}_R$ in $\mathcal{I}_R$
$R_{rd}(t), x_{frd}(t)$	$\mathcal{F}_d$ to $\mathcal{I}_R$ in $\mathcal{I}_R$

Table 1. Coordinate frames relationships.

can be developed:

$$\bar{m}_i = x_f + Rs_{1i} \qquad \bar{m}_{rdi} = x_{frd} + R_{rd}s_{1i} \tag{2}$$

$$\bar{m}_{ri}^* = x_{fr}^* + R_r^* s_{2i} \qquad \bar{m}_i^{'} = x_{fr}^{'} + R^{'}s_{1i} \tag{3}$$

$$\bar{m}_i^* = x_f^* + R^* s_{2i}. \tag{4}$$

In (2)-(4), $\bar{m}_i(t), \bar{m}_i^*(t) \in \mathbb{R}^3$ denote the Euclidean coordinates of the feature points on π and π^*, respectively, expressed in $\mathcal{I}$ as

$$\bar{m}_i(t) \triangleq \begin{bmatrix} x_i(t) & y_i(t) & z_i(t) \end{bmatrix}^T \tag{5}$$

$$\bar{m}_i^*(t) \triangleq \begin{bmatrix} x_i^*(t) & y_i^*(t) & z_i^*(t) \end{bmatrix}^T, \tag{6}$$

$\bar{m}_i^{'}(t), \bar{m}_{rdi}(t) \in \mathbb{R}^3$ denote the actual and desired time-varying Euclidean coordinates, respectively, of the feature points on π expressed in $\mathcal{I}_R$ as

$$\bar{m}_i^{'}(t) \triangleq \begin{bmatrix} x_i^{'}(t) & y_i^{'}(t) & z_i^{'}(t) \end{bmatrix}^T \tag{7}$$

$$\bar{m}_{rdi}(t) \triangleq \begin{bmatrix} x_{rdi}(t) & y_{rdi}(t) & z_{rdi}(t) \end{bmatrix}^T, \tag{8}$$

and $\bar{m}_{ri}^* \in \mathbb{R}^3$ denotes the constant Euclidean coordinates of the feature points on the plane π^* expressed in $\mathcal{I}_R$ as

$$\bar{m}_{ri}^* \triangleq \begin{bmatrix} x_{ri}^* & y_{ri}^* & z_{ri}^* \end{bmatrix}^T. \tag{9}$$

After some algebraic manipulation, the expressions in (2)-(4) can be rewritten as

$$\bar{m}_i^* = \bar{x}_n + R_n \bar{m}_i \tag{10}$$

$$\bar{m}_i = \bar{x}_f + \bar{R}\bar{m}_i^* \qquad \bar{m}_{rdi} = \bar{x}_{frd} + \bar{R}_{rd}\bar{m}_{ri}^* \tag{11}$$

$$\bar{m}_{ri}^* = x_{fr} + R_r \bar{m}_i^* \qquad \bar{m}_i^{'} = x_{fr} + R_r \bar{m}_i, \tag{12}$$

where $R_n(t)$, $\bar{R}(t)$, $\bar{R}_{rd}(t)$, $R_r(t) \in SO(3)$ and $\bar{x}_n(t)$, $\bar{x}_f(t)$, $\bar{x}_{frd}(t)$, $x_{fr}(t) \in \mathbb{R}^3$ are new rotation and translation variables, respectively, defined as[2]

$$\begin{aligned} R_n &= R^* R^T \qquad \bar{R} = R R^{*T} \\ \bar{R}_{rd} &= R_{rd} R_r^{*T} \qquad R_r = R_r^* R^{*T} \end{aligned} \tag{13}$$

$$\bar{x}_n = x_f^* - R_n \left(x_f - R(s_{2i} - s_{1i}) \right) \tag{14}$$

$$\bar{x}_f = x_f - \bar{R} \left(x_f^* + R^* (s_{2i} - s_{1i}) \right) \tag{15}$$

$$\bar{x}_{frd} = x_{frd} - \bar{R}_{rd} \left(x_{fr}^* + R_r^* (s_{2i} - s_{1i}) \right) \tag{16}$$

$$x_{fr} = x_{fr}^* - R_r x_f^* = x'_{fr} - R_r x_f. \tag{17}$$

To facilitate the development of a relationship between the actual Euclidean translation of $\mathcal{F}$ to the Euclidean translation that is reconstructed from the image information, projective relationships are developed from Fig. 1 as

$$d(t) = n^T \bar{m}_i \qquad d^*(t) = n^{*T} \bar{m}_i^* \qquad d_r^* = n_r^{*T} \bar{m}_{ri}^*, \tag{18}$$

where $d(t) \in \mathbb{R}$ represents the distance from the origin of $\mathcal{I}$ to π along the unit normal (expressed in $\mathcal{I}$) to π denoted as $n(t) \in \mathbb{R}^3$, $d^*(t) \in \mathbb{R}$ represents the distance from the origin of $\mathcal{I}$ to π^* along the unit normal (expressed in $\mathcal{I}$) to π^* denoted as $n^*(t) \in \mathbb{R}^3$, and $d_r^* \in \mathbb{R}$ represents the distance from the origin of $\mathcal{I}_R$ to π^* along the unit normal (expressed in $\mathcal{I}_R$) to π^* denoted as $n_r^* \in \mathbb{R}^3$ where $n^*(t) = R_r^T(t) n_r^*$. In (18), $d(t), d^*(t), d_r^* > \varepsilon$ for some positive constant $\varepsilon \in \mathbb{R}$. Based on (18), the relationships in (10)-(12) can be expressed as

$$\bar{m}_i^* = \left(R_n + \frac{\bar{x}_n}{d} n^T \right) \bar{m}_i \qquad \bar{m}_i = \left(\bar{R} + \frac{\bar{x}_f}{d^*} n^{*T} \right) \bar{m}_i^* \tag{19}$$

$$\bar{m}_{rdi} = \left(\bar{R}_{rd} + \frac{\bar{x}_{frd}}{d_r^*} n_r^{*T} \right) \bar{m}_{ri}^* \qquad \bar{m}_{ri}^* = \left(R_r + \frac{x_{fr} n^{*T}}{d^*} \right) \bar{m}_i^* \tag{20}$$

$$\bar{m}_i' = \left(R_r + \frac{x_{fr} n^T}{d} \right) \bar{m}_i. \tag{21}$$

As in Chen et al. (2005), the subsequent development requires that the constant rotation matrix R_r^* be known. The constant rotation matrix R_r^* can be obtained a priori using various methods (e.g., a second camera, additional on-board sensors, off-line calibration, Euclidean measurements). The subsequent development also assumes that the difference between the Euclidean distances $(s_{2i} - s_{1i})$ is a constant $\forall i = 1, ..., n$. While there are many practical applications that satisfy this assumption (e.g., a simple scenario is that the objects attached to $\mathcal{F}$ and $\mathcal{F}^*$ are the identical objects), the assumption is generally restrictive and is the focus of future research. As described in our preliminary work in Hu, Gans, Mehta & Dixon (2007),

[2] Note that $R_n(t)$, $\bar{R}(t)$ and $\bar{R}_{rd}(t)$ in (13) are the rotation matrices between $\mathcal{F}$ and $\mathcal{F}^*$, $\mathcal{F}^*$ and $\mathcal{F}$, and $\mathcal{F}^*$ and $\mathcal{F}_d$, respectively, but $\bar{x}_n(t)$, $\bar{x}_f(t)$ and $\bar{x}_{frd}(t)$ in (14)-(16) are not the translation vectors between the corresponding coordinate frames. Only the rotation matrices will be used in the controller development.

each of these assumptions can be avoided by using the geometric reconstruction approach in Dupree et al. (2007); Gans et al. (2008); Mackunis et al. (2007) under an alternative assumption that the Euclidean distance between two feature points is precisely known.

2.3 Euclidean Reconstruction

The relationships given by (19)-(21) provide a means to quantify a translation and rotation error between the different coordinate systems. Since the pose of $\mathcal{F}$, $\mathcal{F}_d$, and $\mathcal{F}^*$ cannot be directly measured, a Euclidean reconstruction is developed to obtain the pose error by comparing multiple images acquired from the hovering monocular vision system. To facilitate the subsequent development, the normalized Euclidean coordinates of the feature points in π and π^* can be expressed in terms of $\mathcal{I}$ as $m_i(t)$, $m_i^*(t) \in \mathbb{R}^3$, respectively, as

$$m_i \triangleq \frac{\bar{m}_i}{z_i} \qquad m_i^* \triangleq \frac{\bar{m}_i^*}{z_i^*}. \tag{22}$$

Similarly, the normalized Euclidean coordinates of the feature points in π, π_d and π^* can be expressed in terms of $\mathcal{I}_R$ as $m_i^{'}(t)$, $m_{rdi}(t)$, $m_{ri}^* \in \mathbb{R}^3$, respectively, as

$$m_i^{'}(t) \triangleq \frac{\bar{m}_i^{'}(t)}{z_i^{'}(t)} \quad m_{rdi}(t) \triangleq \frac{\bar{m}_{rdi}(t)}{z_{rdi}(t)} \quad m_{ri}^* \triangleq \frac{\bar{m}_{ri}^*}{z_{ri}^*}. \tag{23}$$

From the expressions given in (19) and (22), the rotation and translation between the coordinate systems $\mathcal{F}$ and $\mathcal{F}^*$, between $\mathcal{F}^*$ and $\mathcal{F}_d$, and between $\mathcal{I}$ and $\mathcal{I}_R$ can now be related in terms of the normalized Euclidean coordinates as

$$m_i = \alpha_i \left(\bar{R} + x_h n^{*T}\right) m_i^* \qquad m_i^* = \frac{1}{\alpha_i}\left(R_n + x_{nh} n^T\right) m_i \tag{24}$$

$$m_{rdi} = \alpha_{rdi}\left(\bar{R}_{rd} + x_{hrd} n_r^{*T}\right) m_{ri}^* \qquad m_{ri}^* = \alpha_{ri}\left(R_r + x_{hr} n^{*T}\right) m_i^* \tag{25}$$

where $\alpha_i(t)$, $\alpha_{rdi}(t)$, $\alpha_{ri}(t) \in \mathbb{R}$ denote depth ratios defined as

$$\alpha_i = \frac{z_i^*}{z_i} \qquad \alpha_{rdi} = \frac{z_{ri}^*}{z_{rdi}} \qquad \alpha_{ri} = \frac{z_i^*}{z_{ri}^*},$$

and $x_h(t)$, $x_{nh}(t)$, $x_{hrd}(t)$, $x_{hr}(t) \in \mathbb{R}^3$ denote scaled translation vectors that are defined as

$$x_h = \frac{\bar{x}_f}{d^*} \qquad x_{nh} = \frac{\bar{x}_n}{d} \qquad x_{hrd} = \frac{\bar{x}_{frd}}{d_r^*} \qquad x_{hr} = \frac{x_{fr}}{d^*}. \tag{26}$$

Since the normalized Euclidean coordinates in (24)-(25) can not be directly measured, the following relationships (i.e., the pin-hole camera model) are used to determine the normalized Euclidean coordinates from pixel information:

$$p_i = A_1 m_i \qquad p_i^* = A_1 m_i^* \qquad p_{rdi} = A_2 m_{rdi} \qquad p_{ri}^* = A_2 m_{ri}^*, \tag{27}$$

where $A_1, A_2 \in \mathbb{R}^{3\times 3}$ are known, constant, and invertible intrinsic camera calibration matrices of the current camera and the reference camera, respectively. In (27), $p_i(t)$, $p_i^*(t) \in \mathbb{R}^3$ represent the image-space coordinates of the Euclidean feature points on π and π^* expressed in terms of $\mathcal{I}$ as

$$p_i \triangleq \begin{bmatrix} u_i & v_i & 1 \end{bmatrix}^T \qquad p_i^* \triangleq \begin{bmatrix} u_i^* & v_i^* & 1 \end{bmatrix}^T, \tag{28}$$

respectively, where $u_i(t)$, $v_i(t)$, $u_i^*(t)$, $v_i^*(t) \in \mathbb{R}$. Similarly, $p_{rdi}(t)$, $p_{ri}^* \in \mathbb{R}^3$ represent the image-space coordinates of the Euclidean features on π_d and π^* expressed in terms of $\mathcal{I}_R$ as

$$p_{rdi} \triangleq \begin{bmatrix} u_{rdi} & v_{rdi} & 1 \end{bmatrix}^T \qquad p_{ri}^* \triangleq \begin{bmatrix} u_{ri}^* & v_{ri}^* & 1 \end{bmatrix}^T \tag{29}$$

respectively, where $u_{rdi}(t)$, $v_{rdi}(t)$, u_{ri}^*, $v_{ri}^* \in \mathbb{R}$. By using (24)-(25) and (29), the following relationships can be developed:

$$p_i = \alpha_i \underbrace{\left(A_1\left(\bar{R} + x_h n^{*T}\right)A_1^{-1}\right)}_{G} p_i^* \qquad p_i^* = \frac{1}{\alpha_i}\underbrace{\left(A_1\left(R_n + x_{nh} n^T\right)A_1^{-1}\right)}_{G_n} p_i \tag{30}$$

$$p_{rdi} = \alpha_{rdi} \underbrace{\left(A_2\left(\bar{R}_{rd} + x_{hrd} n_r^{*T}\right)A_2^{-1}\right)}_{G_{rd}} p_{ri}^* \qquad p_{ri}^* = \alpha_{ri}\underbrace{\left(A_2\left(R_r + x_{hr} n^{*T}\right)A_1^{-1}\right)}_{G_r} p_i^*, \tag{31}$$

where $G(t)$, $G_n(t)$, $G_{rd}(t)$, $G_r(t) \in \mathbb{R}^{3\times 3}$ denote projective homographies. Sets of linear equations can be developed from (30)-(31) to determine the projective homographies up to a scalar multiple. Various techniques can be used (e.g., see Faugeras & Lustman (1988); Zhang & Hanson (1995)) to decompose the Euclidean homographies, to obtain $\alpha_i(t)$, $\alpha_{rdi}(t)$, $\alpha_{ri}(t)$, $x_h(t)$, $x_{nh}(t)$, $x_{hrd}(t)$, $x_{hr}(t)$, $\bar{R}(t)$, $R_n(t)$, $\bar{R}_{rd}(t)$, $R_r(t)$, $n^*(t)$, n_r^*, $n(t)$. Given that the constant rotation matrix R_r^* is assumed to be known, the expressions for $\bar{R}_{rd}(t)$ and $R_r(t)$ in (13) can be used to determine $R_{rd}(t)$ and $R^*(t)$. Once $R^*(t)$ is determined, the expression for $\bar{R}(t)$ in (13) can be used to determine $R(t)$. Also, once R_r^*, $R^{*T}(t)$, and $R(t)$ have been determined, (1) can be used to determine $R'(t)$. Since $R_r(t)$, $x_{hr}(t)$, $\alpha_i(t)$, $n^*(t)$, n_r^*, $n(t)$, $m_i^*(t)$, and $m_i(t)$ can be determined, the following relationship can be used to determine $m_i'(t)$:

$$m_i' = \frac{z_i}{z_i'}\left(R_r + x_{hr}\alpha_i \frac{n^{*T} m_i^*}{n^T m_i} n^T\right) m_i, \tag{32}$$

where the inverse of the ratio $\dfrac{z_i(t)}{z_i'(t)}$ can be determined as

$$\frac{z_i'}{z_i} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}\left(R_r + x_{hr}\alpha_i \frac{n^{*T} m_i^*}{n^T m_i} n^T\right) m_i. \tag{33}$$

2.4 Control Objective

The control objective is for a controlled agent (e.g., an UGV or an UAV) to track a desired trajectory that is determined by a sequence of images. This objective is based on the assumption that the control agent is physically able to follow the desired image trajectory, that the linear and angular velocities of the camera are control inputs that can be independently controlled (i.e., unconstrained motion), and that the reference and desired cameras are calibrated (i.e., A_1 and A_2 are known). The control objective can be stated as the desire for the Euclidean feature points on π to track the corresponding feature points on π_d, which can be mathematically stated as the desire for $\bar{m}_i'(t) \to \bar{m}_{rdi}(t)$. Equivalently, the control objective can also be stated in terms of the rotation and translation of the agent as the desire for $x_{fr}'(t) \to x_{frd}(t)$ and $R'(t) \to R_{rd}(t)$.

As stated previously, $R'(t)$ and $R_{rd}(t)$ can be computed by decomposing the projective homographies in (30)-(31) and using (1). Once these rotation matrices have been determined, a variety of parameterizations can be used to describe the rotation. The unit quaternion parameterization is used to describe the rotation in the subsequent problem formulation, control development, and stability analysis since the unit quaternion provides a globally nonsingular parameterization of the corresponding rotation matrices.
The unit quaternion is a four dimensional vector, which can be defined as

$$q \triangleq \begin{bmatrix} q_0 & q_v^T \end{bmatrix}^T \qquad q_v \triangleq \begin{bmatrix} q_{v1} & q_{v2} & q_{v3} \end{bmatrix}^T, \tag{34}$$

where $q_0(t),\ q_{vi}(t) \in \mathbb{R}\ \forall i = 1,2,3$ satisfy the following nonlinear constraint

$$q^T q = 1. \tag{35}$$

Given the rotation matrices $R'(t)$ and $R_{rd}(t)$, the corresponding unit quaternions $q(t)$ and $q_d(t)$ can be calculated by using the numerically robust method presented in Hu et al. (2006) and Shuster (1993) based on the corresponding relationships

$$R' = \left(q_0^2 - q_v^T q_v\right) I_3 + 2q_v q_v^T + 2q_0 q_v^\times \tag{36}$$

$$R_{rd} = \left(q_{0d}^2 - q_{vd}^T q_{vd}\right) I_3 + 2q_{vd} q_{vd}^T + 2q_{0d} q_{vd}^\times \tag{37}$$

where I_3 is the 3×3 identity matrix, and the notation $q_v^\times(t)$ denotes a skew-symmetric form of the vector $q_v(t)$ as

$$q_v^\times = \begin{bmatrix} 0 & -q_{v3} & q_{v2} \\ q_{v3} & 0 & -q_{v1} \\ -q_{v2} & q_{v1} & 0 \end{bmatrix}, \ \forall q_v = \begin{bmatrix} q_{v1} \\ q_{v2} \\ q_{v3} \end{bmatrix}. \tag{38}$$

To quantify the rotation error between the feature points on π and π_d, the multiplicative error between rotation matrices $R'(t)$ and $R_{rd}(t)$ is defined as

$$\tilde{R} = R'^T R_{rd} = \left(\tilde{q}_0^2 - \tilde{q}_v^T \tilde{q}_v\right) I_3 + 2\tilde{q}_v \tilde{q}_v^T - 2\tilde{q}_0 \tilde{q}_v^\times, \tag{39}$$

where the error quaternion $\tilde{q}(t) = (\tilde{q}_0(t), \tilde{q}_v^T(t))^T$ is defined as

$$\tilde{q} = \begin{bmatrix} \tilde{q}_0 \\ \tilde{q}_v \end{bmatrix} = \begin{bmatrix} q_0 q_{0d} + q_v^T q_{vd} \\ q_{0d} q_v - q_0 q_{vd} + q_v^\times q_{vd} \end{bmatrix}. \tag{40}$$

Since $\tilde{q}(t)$ is a unit quaternion, (39) can be used to quantify the rotation tracking objective as

$$\|\tilde{q}_v(t)\| \to 0 \Longrightarrow \tilde{R}(t) \to I_3 \quad \text{as} \quad t \to \infty. \tag{41}$$

The translation error, denoted by $e(t) \in \mathbb{R}^3$, is defined as

$$e = m_e - m_{ed} \tag{42}$$

where $m_e(t), m_{ed}(t) \in \mathbb{R}^3$ are defined as[3]

$$m_e = \begin{bmatrix} \frac{x_1'}{z_1'} & \frac{y_1'}{z_1'} & \ln(\frac{z_1'}{z_{r1}^*}) \end{bmatrix}^T \qquad m_{ed} = \begin{bmatrix} \frac{x_{rd1}}{z_{rd1}} & \frac{y_{rd1}}{z_{rd1}} & \ln(\frac{z_{rd1}}{z_{r1}^*}) \end{bmatrix}^T. \tag{43}$$

In (43), $\frac{z_1'(t)}{z_{r1}^*(t)}$ and $\frac{z_{rd1}(t)}{z_{r1}^*(t)}$ can be expressed in terms of known signals as

$$\frac{z_1'}{z_{r1}^*} = \frac{z_1'}{z_1}\frac{z_1}{z_1^*}\frac{z_1^*}{z_{r1}^*} = \frac{z_1'}{z_1}\frac{1}{\alpha_1}\alpha_{r1} \qquad \frac{z_{rd1}}{z_{r1}^*} = \frac{1}{\alpha_{rd1}}.$$

Based on (41) and (42), the subsequent control development targets the following objectives:

$$\|\tilde{q}_v(t)\| \to 0 \quad \text{and} \quad \|e(t)\| \to 0 \quad \text{as} \quad t \to \infty. \tag{44}$$

2.5 Control Development

2.5.1 Open-Loop Error System

Based on (39) and (40), the open-loop rotation error system can be developed as Dixon et al. (2003)

$$\dot{\tilde{q}} = \frac{1}{2}\begin{bmatrix} -\tilde{q}_v^T \\ \tilde{q}_0 I_3 + \tilde{q}_v^\times \end{bmatrix}\left(\omega_c - \tilde{R}\omega_{cd}\right), \tag{45}$$

where $\omega_{cd}(t)$ denotes the angular velocity of π_d expressed in $\mathcal{F}_d$ that can be calculated as Dixon et al. (2003)

$$\omega_{cd} = 2(q_{0d}\dot{q}_{vd} - q_{vd}\dot{q}_{0d}) - 2q_{vd}^\times \dot{q}_{vd}, \tag{46}$$

where $\left(q_{0d}(t), q_{vd}^T(t)\right)^T$, $\left(\dot{q}_{0d}(t), \dot{q}_{vd}^T(t)\right)^T$ are assumed to be bounded; hence, $\omega_{cd}(t)$ is also bounded. The open-loop translation error system can be derived as

$$z_{r1}^*\dot{e} = \frac{z_{r1}^*}{z_1'}L_v'R'\left(v_c + \omega_c^\times s_1\right) - z_{r1}^*\dot{m}_{ed}, \tag{47}$$

where $v_c(t), \omega_c(t) \in \mathbb{R}^3$ denote the linear and angular velocity of π expressed in $\mathcal{F}$, respectively, and the auxiliary measurable term $L_v'(t) \in \mathbb{R}^{3\times3}$ is defined as

$$L_v' = \begin{bmatrix} 1 & 0 & -\frac{x_1'}{z_1'} \\ 0 & 1 & -\frac{y_1'}{z_1'} \\ 0 & 0 & 1 \end{bmatrix}.$$

[3] Any point O_i can be utilized in the subsequent development; however, to reduce the notational complexity, we have elected to select the image point O_1, and hence, the subscript 1 is utilized in lieu of i in the subsequent development.

2.5.2 Closed-Loop Error System

Based on the open-loop rotation error system in (45) and the subsequent Lyapunov-based stability analysis, the angular velocity controller is designed as

$$\omega_c = -K_\omega \tilde{q}_v + \tilde{R}\omega_{cd}, \tag{48}$$

where $K_\omega \in \mathbb{R}^{3\times 3}$ denotes a diagonal matrix of positive constant control gains. From (45) and (48), the rotation closed-loop error system can be determined as

$$\dot{\tilde{q}}_0 = \frac{1}{2}\tilde{q}_v^T K_\omega \tilde{q}_v \qquad \dot{\tilde{q}}_v = -\frac{1}{2}\left(\tilde{q}_0 I_3 + \tilde{q}_v^\times\right) K_\omega \tilde{q}_v = -\frac{1}{2}K_\omega \tilde{q}_0 \tilde{q}_v. \tag{49}$$

Based on (47), the translation control input $v_c(t)$ is designed as

$$v_c = -\frac{z_1'}{z_{r1}^*} R'^T L_v'^{-1}\left(K_v e - \hat{z}_{r1}^* \dot{m}_{ed}\right) - \omega_c^\times s_1, \tag{50}$$

where $K_v \in \mathbb{R}^{3\times 3}$ denotes a diagonal matrix of positive constant control gains. In (50), the parameter estimate $\hat{z}_{r1}^*(t) \in \mathbb{R}$ for the unknown constant z_{r1}^* is designed as

$$\dot{\hat{z}}_{r1}^* = -\gamma e^T \dot{m}_{ed}, \tag{51}$$

where $\gamma \in \mathbb{R}$ denotes a positive constant adaptation gain. By using (47) and (50), the translation closed-loop error system is

$$z_{r1}^* \dot{e} = -K_v e - \tilde{z}_{r1}^* \dot{m}_{ed}, \tag{52}$$

where $\tilde{z}_{r1}^*(t) \in \mathbb{R}$ denotes the parameter estimation error

$$\tilde{z}_{r1}^* = z_{r1}^* - \hat{z}_{r1}^*. \tag{53}$$

2.5.3 Stability Analysis

Theorem 1. *The controller given in (48) and (50), along with the adaptive update law in (51) ensures asymptotic tracking in the sense that*

$$\|\tilde{q}_v(t)\| \to 0, \ \ \|e(t)\| \to 0, \quad as \quad t \to \infty. \tag{54}$$

Proof. Let $V(t) \in \mathbb{R}$ denote the following differentiable non-negative function (i.e., a Lyapunov candidate):

$$V = \tilde{q}_v^T \tilde{q}_v + (1-\tilde{q}_0)^2 + \frac{z_{r1}^*}{2} e^T e + \frac{1}{2\gamma}\tilde{z}_{r1}^{*2}. \tag{55}$$

The time-derivative of $V(t)$ can be determined as

$$\begin{aligned}
\dot{V} &= -\tilde{q}_v^T K_\omega \tilde{q}_0 \tilde{q}_v - (1-\tilde{q}_0)\tilde{q}_v^T K_\omega \tilde{q}_v - e^T K_v e \\
&\quad + e^T\left(-K_v e - \tilde{z}_{r1}^* \dot{m}_{ed}\right) + \tilde{z}_{r1}^* e^T \dot{m}_{ed} \\
&= -\tilde{q}_v^T\left(\tilde{q}_0 I_3 + (1-\tilde{q}_0) I_3\right) K_\omega \tilde{q}_v - e^T K_v e \\
&= -\tilde{q}_v^T K_\omega \tilde{q}_v - e^T K_v e,
\end{aligned} \tag{56}$$

where (49) and (51)-(53) were utilized. Based on (55) and (56), $e(t)$, $\tilde{q}_v(t)$, $\tilde{q}_0(t)$, $\tilde{z}^*_{r1}(t) \in \mathcal{L}_\infty$ and $e(t)$, $\tilde{q}_v(t) \in \mathcal{L}_2$. Since $\tilde{z}^*_{r1}(t) \in \mathcal{L}_\infty$, it is clear from (53) that $\hat{z}^*_{r1}(t) \in \mathcal{L}_\infty$. Based on the fact that $e(t) \in \mathcal{L}_\infty$, (42) and (43) can be used to prove that $m'_1(t) \in \mathcal{L}_\infty$, and then $L'_v(t)$, $L'^{-1}_v(t) \in \mathcal{L}_\infty$. Based on the fact that $\tilde{q}_v(t) \in \mathcal{L}_\infty$ and $\omega_{cd}(t)$ is a bounded function, (48) can be used to conclude that $\omega_c(t) \in \mathcal{L}_\infty$. Since $\hat{z}^*_{r1}(t)$, $e(t)$, $m'_1(t)$, $L'_v(t)$, $L'^{-1}_v(t) \in \mathcal{L}_\infty$ and $\dot{m}_{ed}(t)$ is bounded, (50) can be utilized to prove that $v_c(t) \in \mathcal{L}_\infty$. From the previous results, (45)-(47) can be used to prove that $\dot{e}(t)$, $\dot{\tilde{q}}_v(t) \in \mathcal{L}_\infty$. Since $e(t)$, $\tilde{q}_v(t) \in \mathcal{L}_\infty \cap \mathcal{L}_2$, and $\dot{e}(t)$, $\dot{\tilde{q}}_v(t) \in \mathcal{L}_\infty$, we can utilize a corollary to Barbalat's Lemma Slotine & Li (1991) to conclude the result given in (54). □

3. Cooperative Tracking Control of A Nonholonomic Unmanned Ground Vehicle

In the previous section, a visual servo tracking controller is developed for a moving six-DOF agent based on daisy-chained image feedback from a moving camera where a stationary reference camera was used to encode a desired video. The development in this section and our preliminary work in Mehta, Hu, Gans & Dixon (2006) extends the previous section by allowing the reference camera to also move. The example of a reference camera in the previous section was a "stationary" satellite that was used to encode the desired trajectory. In this section, the desired trajectory could be encoded by a moving camera (e.g., attached to a moving satellite, a dirigible, or another UAV). In addition, instead of considering the general six-DOF control agent, the control agent in this section is a nonholonomic constrained UGV. The control objective is for the UGV to track a desired trajectory determined by a sequence of prerecorded images from some moving overhead camera. An additional technical issue resolved in this section is the challenge of comparing the relative velocity between a moving camera and a moving UGV to the relative desired trajectory recorded by a moving camera.

3.1 Problem Scenario

Previous results, such as Chen et al. (2006); Dixon et al. (2001); Fang et al. (2005), are typically focused on the results in which the targets are static with respect to the moving camera or the camera is stationary and recording images of the moving UGV. In contrast to these methods, the development in this section is motivated by the problem when a moving camera is recording images of a moving UGV so a second UGV can track a desired image trajectory.

The scenario examined in this section is depicted in Fig. 2, where various coordinate frames are defined again as a means to develop the subsequent Euclidean reconstruction and control methods. In Fig. 2, a single camera is navigating above the motion plane of an UGV. The moving coordinate frame $\mathcal{I}$ is attached to an overhead camera, which records an images for real-time tracking control. The moving coordinate frame $\mathcal{I}_M$ is attached to the overhead camera that recorded the desired image sequence, and the fixed coordinate frame $\mathcal{I}_R$ is some single snapshot of $\mathcal{I}_M$.

The moving coordinate frame $\mathcal{F}$ is attached to the UGV at the center of the rear wheel axis (for simplicity and without loss of generality). The UGV is represented in the camera image by four feature points that are coplanar and not collinear. The Euclidean distance (i.e., $s_{1i} \in \mathbb{R}^3$ $\forall i = 1, 2, 3, 4$) from the origin of $\mathcal{F}$ to one of the feature points is assumed to be known. A priori information (such as a known target in the initial FOV Davison et al. (2007)) is sometimes used in VSLAM methods to establish scale. The plane defined by the UGV motion (i.e., the plane defined by the xy-axis of $\mathcal{F}$) and the UGV feature points is denoted by π. The linear

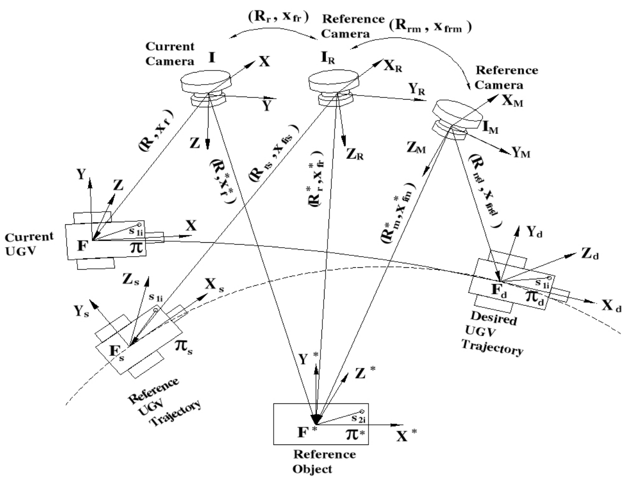


Fig. 2. Geometric model for a moving camera, moving target and moving reference camera.

velocity of the UGV along the x-axis of $\mathcal{F}$ is denoted by $v_c(t) \in \mathbb{R}$, and the angular velocity $\omega_c(t) \in \mathbb{R}$ is about the z-axis of $\mathcal{F}$.
While viewing the feature points of the UGV, the camera is assumed to also view four additional coplanar and noncollinear feature points of a stationary reference object. The four additional feature points define the plane π^* in Fig. 2. The stationary coordinate frame $\mathcal{F}^*$ is attached to the object where a distance from the origin of the coordinate frame to one of the feature points (i.e., $s_{2i} \in \mathbb{R}^3$) is assumed to be known. The plane π^* is assumed to be parallel to the plane π. When the camera is coincident with $\mathcal{I}_R$, a fixed (i.e., a single snapshot) reference pose of the UGV, denoted by $\mathcal{F}_s$, is assumed to be in the camera's FOV. A desired trajectory is defined by a prerecorded time-varying trajectory of $\mathcal{F}_d$ that is assumed to be second-order differentiable where $v_{cd}(t), \omega_{cd}(t) \in \mathbb{R}$ denote the desired linear and angular velocity of $\mathcal{F}_d$, respectively. The feature points that define π^* are also assumed to be visible when the camera is a priori located coincident with the pose of the stationary coordinate frame $\mathcal{I}_R$ and the time-varying coordinate frame $\mathcal{I}_M$. Based on these coordinate frame definitions, the problem considered in this section is to develop a kinematic controller for the object attached to $\mathcal{F}$ so that the time-varying rotation and translation of $\mathcal{F}$ converges to the desired time-varying rotation and translation of $\mathcal{F}_d$, where the motion of $\mathcal{F}$ is determined from the time-varying overhead camera attached to $\mathcal{I}$.

3.2 Geometric Relationships

The rotation matrices and translation vectors in Table I (except the last line) are also valid for this section. Additional relationships between the various coordinate frames are summarized in Table II. In Table II, R_{rs}, $R_{md}(t)$, $R_m^*(t)$, $R_{rm}(t)$, $R_{md}^{'}(t) \in SO(3)$ denote rotation matrices, and x_{frs}, $x_{fmd}(t)$, $x_{fm}^*(t)$, $x_{frm}(t)$, $x_{frm}'(t) \in \mathbb{R}^3$ denote translation vectors.

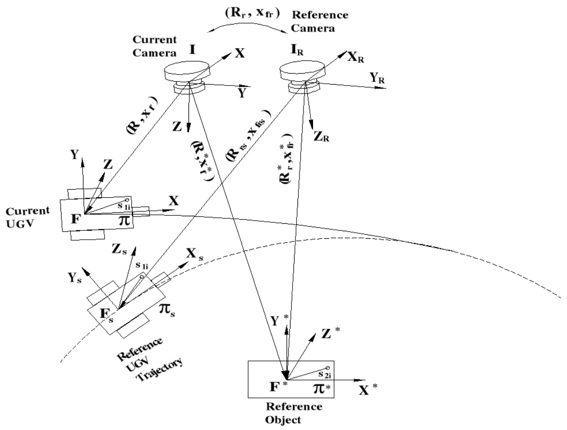


Fig. 3. Geometric model for a moving camera, moving target and a stationary coordinate frame attached to a snapshot of the moving reference camera.

Motion	Frames
R_{rs}, x_{frs}	$\mathcal{F}_s$ to $\mathcal{I}_R$
$R_{md}(t), x_{fmd}(t)$	$\mathcal{F}_d$ to $\mathcal{I}_M$
$R_m^*(t), x_{fm}^*(t)$	$\mathcal{F}^*$ to $\mathcal{I}_M$
$R_{rm}(t), x_{frm}(t)$	$\mathcal{I}_M$ to $\mathcal{I}_R$
$R'_{md}(t), x'_{frm}(t)$	$\mathcal{F}$ to $\mathcal{I}_R$ in $\mathcal{I}_M$

Table 2. Coordinate frame relationships.

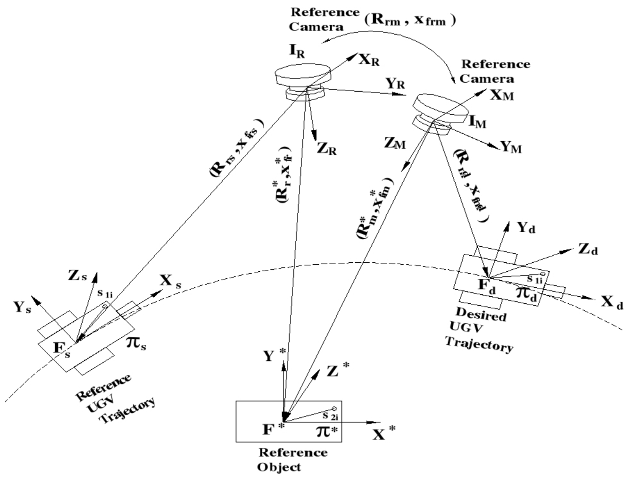


Fig. 4. Geometric model for a moving target, moving reference camera, desired trajectory, and a stationary coordinate frame attached to a snapshot of the moving reference camera.

3.3 Euclidean Reconstruction

The coordinate frame representation in Fig. 2 can be separated into Figs. 3 and 4 to relate $\mathcal{I}$ to $\mathcal{I}_R$ and $\mathcal{I}_R$ to $\mathcal{I}_M$, respectively. The coordinate frames in each figure have the same relationships as in Fig. 1. Therefore, the same Euclidean reconstruction process as presented in Section 2.1-2.3 can be used twice to build the Euclidean relationships for this example.
To reconstruct the Euclidean relationship for the geometric model as shown in Fig. 3, let $\bar{m}_{rsi} \in \mathbb{R}^3$ denote the constant reference Euclidean coordinates of the feature points on π_s expressed in $\mathcal{I}_R$ as

$$\bar{m}_{rsi} \triangleq \begin{bmatrix} x_{rsi} & y_{rsi} & z_{rsi} \end{bmatrix}^T,$$

and let $p_{rsi} \in \mathbb{R}^3$ represent the constant image-space coordinates of the feature points on π_s taken by the camera attached to $\mathcal{I}_M$ when $\mathcal{I}_M$ is coincident with $\mathcal{I}_R$

$$p_{rsi} \triangleq \begin{bmatrix} u_{rsi} & v_{rsi} & 1 \end{bmatrix}^T.$$

Following the development in Section 2.2 and 2.3, relationships can be obtained to determine the homographies and depth ratios as[4]

$$p_i = \alpha_i \underbrace{\left(A\left(\bar{R} + x_h n^{*T}\right)A^{-1}\right)}_{G} p_i^* \qquad p_{rsi} = \alpha_{rsi} \underbrace{\left(A\left(\bar{R}_{rs} + x_{hrs} n_r^{*T}\right)A^{-1}\right)}_{G_{rs}} p_{ri}^* \tag{57}$$

$$p_{ri}^* = \alpha_{ri} \underbrace{\left(A\left(R_r + x_{hr} n^{*T}\right)A^{-1}\right)}_{G_r} p_i^* \tag{58}$$

where

$$\begin{aligned} \alpha_i &= \frac{z_i^*}{z_i} \qquad \alpha_{rsi} = \frac{z_{ri}^*}{z_{rsi}} \qquad \alpha_{ri} = \frac{z_i^*}{z_{ri}^*} \\ \bar{R} &= RR^{*T} \quad \bar{R}_{rs} = R_{rs}R_r^{*T} \quad R_r = R_r^* R^{*T}. \end{aligned} \tag{59}$$

Furthermore, the normalized Euclidean coordinates $m_i(t)$ can be related to $m_i'(t)$ as

$$m_i' = \frac{z_i}{z_i'}\left(R_r + x_{hr}\alpha_i \frac{n^{*T}m_i^*}{n^{*T}m_i} n^{*T}\right) m_i \tag{60}$$

$$\frac{z_i'}{z_i} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \left(R_r + x_{hr}\alpha_i \frac{n^{*T}m_i^*}{n^T m_i} n^{*T}\right) m_i. \tag{61}$$

To reconstruct the Euclidean relationship for the geometric model as shown in Fig. 4, let $\bar{m}_{mdi}(t)$, $\bar{m}_{mi}^*(t) \in \mathbb{R}^3$ denote the Euclidean coordinates of the feature points on π_d and π^* expressed in $\mathcal{I}_M$ as

$$\begin{aligned} \bar{m}_{mdi}(t) &\triangleq \begin{bmatrix} x_{mdi}(t) & y_{mdi}(t) & z_{mdi}(t) \end{bmatrix}^T \\ \bar{m}_{mi}^*(t) &\triangleq \begin{bmatrix} x_{mi}^*(t) & y_{mi}^*(t) & z_{mi}^*(t) \end{bmatrix}^T, \end{aligned}$$

[4] To simplify the notations, the cameras are assumed to have the same calibration matrix A in the following development. The readers can refer to Section 2.1 for the deductions that the calibration matrices are different.

let $\bar{m}'_{mdi}(t) \in \mathbb{R}^3$ denote the desired Euclidean coordinates of the feature points on π_d expressed in $\mathcal{I}_R$ as

$$\bar{m}'_{mdi}(t) \triangleq \begin{bmatrix} x'_{mdi}(t) & y'_{mdi}(t) & z'_{mdi}(t) \end{bmatrix}^T,$$

and let $p_{mdi}(t)$, $p^*_{mi}(t) \in \mathbb{R}^3$ represent the image-space coordinates of the feature points on π_d and π^* captured by the camera attached to $\mathcal{I}_M$, respectively, as

$$p_{mdi} \triangleq \begin{bmatrix} u_{mdi} & v_{mdi} & 1 \end{bmatrix}^T \qquad p^*_{mi} \triangleq \begin{bmatrix} u^*_{mi} & v^*_{mi} & 1 \end{bmatrix}^T.$$

The normalized coordinates of $\bar{m}'_{mdi}(t)$ and $\bar{m}_{mdi}(t)$, denoted as $m'_{mdi}(t)$, $m_{mdi}(t) \in \mathbb{R}^3$, respectively, are defined as

$$m'_{mdi}(t) \triangleq \frac{\bar{m}'_{mdi}(t)}{z'_{mdi}(t)} \qquad m_{mdi}(t) \triangleq \frac{\bar{m}_{mdi}(t)}{z_{mdi}(t)}. \tag{62}$$

Following the development in Section 2.2 and 2.3, relationships can be developed to compute the homographies and depth ratios as

$$p_{mdi} = \alpha_{mdi} \underbrace{\left(A\left(\bar{R}_{md} + x_{hmd}n_m^{*T}\right)A^{-1}\right)}_{G_{md}} p^*_{mi} \qquad p^*_{ri} = \alpha_{rmi} \underbrace{\left(A\left(R_{rm} + x_{hrm}n_m^{*T}\right)A^{-1}\right)}_{G_{rm}} p^*_{mi}, \tag{63}$$

where

$$\begin{aligned} \alpha_{mdi} &= \frac{z^*_{mi}}{z_{mdi}} \qquad \alpha_{rmi} = \frac{z^*_{mi}}{z^*_{ri}} \\ \bar{R}_{md} &= R_{md}R_m^{*T} \qquad R_{rm} = R^*_r R_m^{*T}. \end{aligned} \tag{64}$$

Relationships between $m_{mdi}(t)$ and $m'_{mdi}(t)$ can be established as

$$m'_{mdi} = \frac{z_{mdi}}{z'_{mdi}} \left(R_{rm} + x_{hrm}\alpha_{mdi}\frac{n_m^{*T}m^*_{mi}}{n_m^{*T}m_{mdi}}n_m^{*T}\right) m_{mdi} \tag{65}$$

$$\frac{z'_{mdi}}{z_{mdi}} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \left(R_{rm} + x_{hrm}\alpha_{mdi}\frac{n_m^{*T}m^*_{mi}}{n_m^{*T}m_{mdi}}n_m^{*T}\right) m_{mdi}. \tag{66}$$

In (57)-(63), $n^*(t)$, $n^*_m(t)$, and $n^*_r \in \mathbb{R}^3$ denote the constant unit normal to the planes π and π^* as expressed in $\mathcal{I}$, $\mathcal{I}_M$, and $\mathcal{I}_R$ respectively, $x_h(t)$, $x_{hrs}(t)$, $x_{hr}(t)$, $x_{hmd}(t)$, $x_{hrm}(t) \in \mathbb{R}^3$ denote the corresponding scaled translation vectors, and $G(t)$, G_{rs}, $G_r(t)$, $G_{md}(t)$, $G_{rm}(t) \in R^{3\times 3}$ denote projective homographies.

Sets of linear equations in (57)-(58) and (63) can be used to determine and decompose homographies to obtain $\alpha_i(t)$, α_{rsi}, $\alpha_{mdi}(t)$, $\alpha_{ri}(t)$, $\alpha_{rmi}(t)$, $x_h(t)$, x_{hrs}, $x_{hr}(t)$, $x_{hmd}(t)$, $x_{hrm}(t)$, $\bar{R}(t)$, $\bar{R}_{rs}$, $R_r(t)$, $\bar{R}_{md}(t)$, and $R_{rm}(t)$. Given that the rotation matrix $R^*_r(t)$ is assumed to be known, the expressions for $\bar{R}_{rs}(t)$ and $R_r(t)$ in (59) can be used to determine $R_{rs}(t)$ and $R^*(t)$. Once $R^*(t)$ is determined, the expression for $\bar{R}(t)$ and $R_{rm}(t)$ in (59) and (64) can be used to determine $R(t)$ and $R^*_m(t)$. The rotation $R^*_m(t)$ can then be used to calculate $R_{md}(t)$ from the relationship for $\bar{R}_{md}$ in (64). Based on the definitions for $R(t)$, $R^*(t)$, $R_{md}(t)$, $R^*_m(t)$, R^*_r, and R_{rs} provided in the previous development, the rotation from $\mathcal{F}$ to $\mathcal{F}_s$ and from $\mathcal{F}_d$ to $\mathcal{F}_s$, denoted by $R_1(t)$, $R_{d1}(t) \in SO(3)$, respectively, are defined as

$$R_1(t) = R_{rs}^T R_r^* R^{*T}(t) R(t) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{67}$$

$$R_{d1}(t) \quad = \quad R_{rs}^T R_r^* R_m^{*T}(t) R_{md}(t) = \begin{bmatrix} \cos\theta_d & \sin\theta_d & 0 \\ -\sin\theta_d & \cos\theta_d & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{68}$$

where $\theta(t) \in \mathbb{R}$ denotes the right-handed rotation angle about the z-axis that aligns $\mathcal{F}$ with $\mathcal{F}_s$, and $\theta_d(t) \in \mathbb{R}$ denotes the right-handed rotation angle about the z-axis that aligns $\mathcal{F}_d$ with $\mathcal{F}_s$. From the definitions of $\theta(t)$ and $\theta_d(t)$, it is clear that

$$\dot{\theta} = \omega_c \qquad \dot{\theta}_d = \omega_{cd} \tag{69}$$

where $\omega_c(t), \omega_{cd}(t) \in \mathbb{R}$ denote the desired angular velocities of $\mathcal{F}$ and $\mathcal{F}_d$, respectively. Based on the fact that $R(t)$, $R^*(t)$, $R_{md}(t)$, $R_m^*(t)$, R_r^*, and R_{rs} are known, it is clear from (67)-(69) that $\theta(t)$ and $\theta_d(t)$ are known signals that can be used in the subsequent control development. To facilitate the subsequent development, $\theta(t)$ and $\theta_d(t)$ are assumed to be confined to the following regions

$$-\pi < \theta(t) \leqslant \pi \qquad -\pi < \theta_d(t) \leqslant \pi. \tag{70}$$

3.4 Control Objective

The objective is to develop a visual servo controller that ensures that the coordinate system $\mathcal{F}$ tracks the time-varying trajectory of $\mathcal{F}_d$ (i.e., $\bar{m}_i(t)$ measured in $\mathcal{I}$ tracks $\bar{m}_{mdi}(t)$ measured in $\mathcal{I}_M$). To ensure that $\bar{m}_i(t)$ tracks $\bar{m}_{mdi}(t)$, the control objective can be stated by using the Euclidean reconstruction given in (57)-(63) as the desire for $\bar{m}_1'(t) \rightarrow \bar{m}_{md1}'(t)$. To quantify the control objective, translation and rotation tracking error vector, denoted by $e(t) \triangleq [e_1(t), e_2(t), e_3(t)]^T \in \mathbb{R}^3$, is defined as Dixon et al. (2003)

$$e_1 \triangleq \eta_1 - \eta_{d1} \qquad e_2 \triangleq \eta_2 - \eta_{d2} \qquad e_3 \triangleq \theta - \theta_d \tag{71}$$

where $\theta(t)$ and $\theta_d(t)$ are introduced in (67) and (68), respectively, and the auxiliary signals $\eta(t) \triangleq [\eta_1(t), \eta_2(t), \eta_3(t)]^T$, $\eta_d(t) \triangleq [\eta_{d1}(t), \eta_{d2}(t), \eta_{d3}(t)]^T \in \mathbb{R}^3$ are defined as

$$\begin{aligned} \eta(t) &\triangleq \frac{1}{z_{r1}^*} R^T(t) R^*(t) R_r^{*T} \bar{m}_1'(t) \\ \eta_d(t) &\triangleq \frac{1}{z_{r1}^*} R_{md}^T(t) R_m^*(t) R_r^{*T} \bar{m}_{md1}'(t). \end{aligned} \tag{72}$$

Also, the normal unit vector n_r^* is defined as Mehta, Hu, Gans & Dixon (2006)

$$\begin{aligned} n_r^* &= R_r^* R^{*T}(t) R(t) \begin{bmatrix} 0 & 0 & -1 \end{bmatrix}^T \\ &= R_r^* R_m^{*T}(t) R_{md}(t) \begin{bmatrix} 0 & 0 & -1 \end{bmatrix}^T. \end{aligned} \tag{73}$$

The expressions in (72) and (73) can be used to determine that

$$\eta_3 = \eta_{d3} = \frac{-d_r}{z_{r1}^*}. \tag{74}$$

The expressions in (57)-(66) can be used to rewrite $\eta(t)$ and $\eta_d(t)$ in terms of the measurable signals $\alpha_1(t)$, $\alpha_{r1}(t)$, $\alpha_{rm1}(t)$, $\alpha_{md1}(t)$, $R(t)$, $R^*(t)$, R_r^*, $R_{md}(t)$, $R_m^*(t)$, $p_1(t)$, and $p_{md1}(t)$ as

$$\begin{aligned} \eta(t) &= \frac{\alpha_{r1}}{\alpha_1} R^T(t) R^*(t) R_r^{*T} H_r' A^{-1} p_1 \\ \eta_d(t) &= \frac{\alpha_{rm1}}{\alpha_{md1}} R_{md}^T(t) R_m^*(t) R_r^{*T} H_{rm}' A^{-1} p_{md1}. \end{aligned} \tag{75}$$

Based on (71), (75), and the fact that $\theta(t)$ and $\theta_d(t)$ are measurable, it is clear that $e(t)$ is measurable. By examining (71)-(74), the control objective is achieved if $\|e(t)\| \to 0$. Specifically, if $e_3(t) \to 0$, then it is clear from (71) that $R_1(t) \to R_{d1}(t)$. If $e_1(t) \to 0$ and $e_2(t) \to 0$, then from (71) and (74) it is clear that $\eta(t) \to \eta_d(t)$. Given that $R_1(t) \to R_{d1}(t)$ and that $\eta(t) \to \eta_d(t)$, then (72) can be used to conclude that $m_1'(t) \to m_{md1}'(t)$. If $m_1'(t) \to m_{md1}'(t)$ and $R_1(t) \to R_{d1}(t)$, then the Euclidean relationships in the geometric model can be used to prove that $\bar{m}_i(t)$ measured in terms of $\mathcal{I} \to \bar{m}_{mdi}(t)$ measured in terms of $\mathcal{I}_M$.

3.5 Control Development

The open-loop error system can be obtained by taking the time derivative of (72) as

$$\dot{\eta} = \frac{v}{z_{r1}^*} + \left[\eta - \frac{s_{11}}{z_{r1}^*}\right]^{\times} \omega \tag{76}$$

where $v(t), \omega(t) \in \mathbb{R}^3$ denote the respective linear and angular velocity of an UGV expressed in $\mathcal{F}$ as

$$v \triangleq \begin{bmatrix} v_c & 0 & 0 \end{bmatrix}^T \qquad \omega \triangleq \begin{bmatrix} 0 & 0 & \omega_c \end{bmatrix}^T. \tag{77}$$

Without loss of generality, the location of the feature point s_1 is taken as the origin of $\mathcal{F}$, so that $s_{11} = [0, 0, 0]^T$. Then, based on (76) and (77), the error system can be further written as

$$\dot{\eta}_1 = \frac{v_c}{z_{r1}^*} + \eta_2 \omega_c \qquad \dot{\eta}_2 = -\eta_1 \omega_c. \tag{78}$$

Since the desired trajectory is assumed to be generated in accordance with UGV motion constraints, a similar expression to (78) can be developed as

$$\dot{\eta}_{d1} = \frac{v_{cd}}{z_{r1}^*} + \eta_{2d} \omega_{cd} \qquad \dot{\eta}_{d2} = -\eta_{d1} \omega_{cd}. \tag{79}$$

where $v_{cd}(t) \in \mathbb{R}$ denotes the desired linear velocity of $\mathcal{F}_d$. From (69), (71), (76) and (78), the open-loop error system can be obtained as

$$\begin{aligned} & z_{r1}^* \dot{e}_1 = v_c + z_{r1}^* (\eta_2 \omega_c - \dot{\eta}_{d1}) \\ & \dot{e}_2 = -\eta_1 \omega_c + \eta_{d1} \dot{\theta}_d \\ & \dot{e}_3 = \omega_c - \dot{\theta}_d. \end{aligned} \tag{80}$$

To facilitate the subsequent development, the auxiliary variable $\bar{e}_2(t) \in \mathbb{R}$ is defined as

$$\bar{e}_2 \triangleq e_2 + \eta_{d1} e_3. \tag{81}$$

After taking the time derivative of (81) and utilizing (80), the following expression is obtained:

$$\dot{\bar{e}}_2 = -e_1 \omega_c + \dot{\eta}_{d1} e_3. \tag{82}$$

Based on (81), it is clear that if $\bar{e}_2(t)$, $e_3(t) \to 0$, then $e_2(t) \to 0$. Based on this observation and the open-loop dynamics given in (82), the following control development is based on the desire to show that $e_1(t), \bar{e}_2(t), e_3(t)$ are asymptotically driven to zero.
Based on the open-loop error systems in (80) and (82), the linear and angular velocity control inputs for an UGV are designed as

$$v_c \triangleq -k_v e_1 + \bar{e}_2 \omega_c - \hat{z}^*_{r1}(\eta_2 \omega_c - \dot{\eta}_{d1}) \tag{83}$$

$$\omega_c \triangleq -k_\omega e_3 + \dot{\theta}_d - \dot{\eta}_{d1} \bar{e}_2 \tag{84}$$

where k_v, $k_\omega \in \mathbb{R}$ denote positive, constant control gains. In (83), the parameter update law $\hat{z}^*_{r1}(t) \in \mathbb{R}$ is generated by the differential equation

$$\dot{\hat{z}}^*_{r1} = \gamma_1 e_1 (\eta_2 \omega_c - \dot{\eta}_{d1}) \tag{85}$$

where $\gamma_1 \in \mathbb{R}$ is a positive, constant adaptation gain. After substituting the kinematic control signals designed in (83) and (84) into (80), the following closed-loop error systems are obtained:

$$\begin{aligned} z^*_{r1} \dot{e}_1 &= -k_v e_1 + \bar{e}_2 \omega_c + \tilde{z}^*_{r1}(\eta_2 \omega_c - \dot{\eta}_{d1}) \\ \dot{\bar{e}}_2 &= -e_1 \omega_c + \dot{\eta}_{d1} e_3 \\ \dot{e}_3 &= -k_\omega e_3 - \dot{\eta}_{d1} \bar{e}_2 \end{aligned} \tag{86}$$

where (82) was utilized, and the depth-related parameter estimation error, denoted by $\tilde{z}^*_{r1}(t) \in \mathbb{R}$, is defined as

$$\tilde{z}^*_{r1} \triangleq z^*_{r1} - \hat{z}^*_{r1}. \tag{87}$$

Theorem 2. *The control input designed in (83) and (84) along with the adaptive update law defined in (85) ensure asymptotic tracking for UGV in the sense that*

$$\|e(t)\| \to 0 \tag{88}$$

provided the time derivative of the desired trajectory satisfies the following condition

$$\dot{\eta}_{d1} \nrightarrow 0. \tag{89}$$

Lyapunov-based analysis methods and Barbalat's lemma can be used to proved Theorem 2 based on a positive definite function $V(t) \in \mathbb{R}$ defined as Mehta, Hu, Gans & Dixon (2006)

$$V \triangleq \frac{1}{2} z^*_{r1} e_1^2 + \frac{1}{2} \bar{e}_2^2 + \frac{1}{2} e_3^2 + \frac{1}{2\gamma_1} \tilde{z}^{*2}_{r1}. \tag{90}$$

4. Simultaneous Tracking, Localization and Mapping

For vision-based autonomous systems applications (e.g., tracking, localization and mapping), the given reference object can leave the camera's FOV while another reference object enters the FOV. In comparison to the single reference object problem presented in Section 3, multiple reference objects are taken into consideration in this section. The daisy-chaining method is further developed to achieve asymptotic tracking of the UGV by mapping each reference object to a global coordinate system. Moreover, the time-varying Euclidean position of the UGV and the stationary position of the reference objects can be localized with respect to the global coordinate system. In addition to achieving the visual servo tracking and localization objectives, the developed method generates data for the SLAM problem.

4.1 Problem Scenario

The geometric model in this section is the same as in Section 3, except that multiple reference objects are taken into consideration. While viewing the feature points of the UGV, the camera is assumed to also view four additional coplanar and noncollinear feature points of a stationary reference object, such that at any instant of time along the camera motion trajectory at least one such reference object is in the FOV and two reference objects are required to be visible for the frames switching from one reference to the other. The four additional feature points define the plane π_j^* in Fig. 5. The stationary coordinate frame $\mathcal{F}_j^*$ $(j = 1, 2, ..., k)$ is attached to the object where distance from the origin of the coordinate frame to one of the feature points is assumed to be known, i.e., $s_{2ji} \in \mathbb{R}^3$ $\forall i = 1, 2, 3, 4$. The plane π_j^* is assumed to be parallel to the plane π. The feature points that define π_1^*, corresponding to a reference object $\mathcal{F}_1^*$ (i.e., $\mathcal{F}_j^*$ corresponding to $j = 1$), are also assumed to be visible when the camera is a priori located coincident with the pose of the stationary coordinate frame $\mathcal{I}_R$. The fixed coordinate frame $\mathcal{I}_R$ is a snapshot of I_M at the time instant that the first reference object π_1^* is visible to the reference camera. The reference object π_1^* is visible to $\mathcal{I}_R$, but the other reference objects $\pi_j^* (j > 1)$ are not.

4.2 Geometric Relationships

In addition to the notations in Tables I and II, more relationships between the various coordinate frames are summarized in Table III. In Table III, $R_j^*(t)$, $R_{rj}^*(t)$, $R_{mj}^*(t) \in SO(3)$ denote rotation matrices and $x_{fj}^*(t)$, x_{frj}^*, $x_{fmj}^*(t) \in \mathbb{R}^3$ denote translation vectors.

Motion	Frames
$R_j^*(t)$, $x_{fj}^*(t)$	$\mathcal{F}_j^*$ to $\mathcal{I}$ in $\mathcal{I}$
$R_{rj}^*(t)$, x_{frj}^*	$\mathcal{F}_j^*$ to $\mathcal{I}_R$ in $\mathcal{I}_R$
$R_{mj}^*(t)$, $x_{fmj}^*(t)$	$\mathcal{F}_j^*$ to $\mathcal{I}_M$ in $\mathcal{I}_M$

Table 3. Coordinate frame relationships.

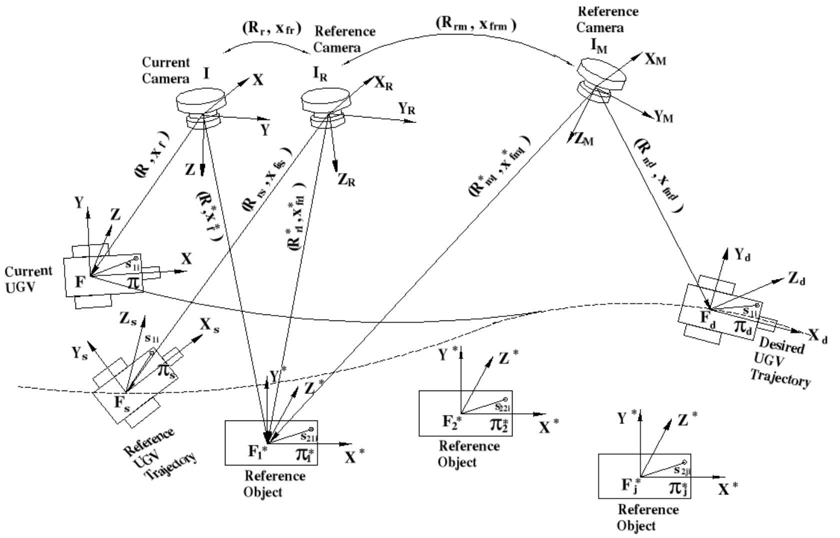


Fig. 5. Geometric model for a moving camera, moving target, moving reference camera and multiple reference objects for simultaneous tracking, localization, and mapping.

4.3 Euclidean Reconstruction

The Euclidean reconstruction for the geometric model in Fig. 5 can be separated into three cases. Case 1: a single reference object π_1^* is within the reference camera's FOV and therefore π_1^* is used as the reference object. Case 2: two reference objects (e.g., π_1^* and π_2^*) are within the camera's FOV, and the reference object in use is going to be switched from one to the other (e.g., from π_1^* to π_2^*). Case 3: π_j^* $(j \geq 2)$ is used as the reference object.

Let $\bar{m}_{mji}^*(t), \bar{m}_{rji}^{'*} \in \mathbb{R}^3$ denote the Euclidean coordinates of the feature points on π_j^* expressed in $\mathcal{I}_M$ and $\mathcal{I}_R$, respectively. Also, let $p_{mji}^*(t), p_{rji}^{'*} \in \mathbb{R}^3$ represent the image-space coordinates of the feature points on π_j^* captured by the reference camera attached to $\mathcal{I}_M$ and $\mathcal{I}_R$, respectively. When $j = 1$, $p_{r1i}^{'*}$ are measurable and denoted by p_{r1i}^*, however, when $j > 1$, $p_{rji}^{'*}$ can not be measured directly and needs to be computed based on the corresponding normalized coordinates obtained from the daisy-chaining multi-view geometry. The normalized coordinates of $\bar{m}_{mji}^*(t)$ and $\bar{m}_{rji}^{'*}$ are denoted as $m_{mji}^*(t), m_{rji}^{'*} \in \mathbb{R}^3$, respectively.

For the first case, the Euclidean reconstruction is exactly the same as that in Section 3. For the second case, consider the feature point planes π_1^* and π_2^* as an example. Similar to the Euclidean reconstruction development in Section 3.3, relationships can be established between

$m'^{*}_{r2i}(t)$ and $m^{*}_{r1i}(t)$ as

$$m'^{*}_{r2i} = \frac{z^{*}_{r1i}}{z'^{*}_{r2i}}\left(R_{21} + x_{h21}\alpha_{rm1i}\frac{n^{*T}_{m1}m^{*}_{m1i}}{n^{*T}_{r1}m^{*}_{r1i}}n^{*T}_{r1}\right)m^{*}_{r1i} \tag{91}$$

$$\frac{z'^{*}_{r2i}}{z^{*}_{r1i}} = \left[\begin{array}{ccc} 0 & 0 & 1 \end{array}\right]\left(R_{21} + x_{h21}\alpha_{rm1i}\frac{n^{*T}_{m1}m^{*}_{m1i}}{n^{*T}_{r1}m^{*}_{r1i}}n^{*T}_{r1}\right)m^{*}_{r1i}. \tag{92}$$

From $m'^{*}_{r2i}(t)$ in (91), the virtual pixel coordinates $p'^{*}_{r2i}(t)$ can be computed. Based on (91) and (92), the Euclidean coordinates of the feature points on π^{*}_{2} can be related to fixed coordinate frame $\mathcal{I}_R$. Following the same idea used to relate π^{*}_{2} and π^{*}_{1}, π^{*}_{j} can be related to π^{*}_{j-1} $(j = 3, ..., k)$ as

$$m'^{*}_{rji} = \frac{z^{*}_{r(j-1)i}}{z'^{*}_{rji}}\left(R_{j(j-1)} + x_{hj(j-1)}\alpha_{rm(j-1)i}\frac{n^{*T}_{m(j-1)}m^{*}_{m(j-1)i}}{n^{*T}_{r(j-1)}m^{*}_{r(j-1)i}}n^{*T}_{r(j-1)}\right)m^{*}_{r(j-1)i} \tag{93}$$

$$\frac{z'^{*}_{rji}}{z^{*}_{r(j-1)i}} = \left[\begin{array}{ccc} 0 & 0 & 1 \end{array}\right]\left(R_{j(j-1)} + x_{hj(j-1)}\alpha_{rm(j-1)i}\frac{n^{*T}_{m(j-1)}m^{*}_{m(j-1)i}}{n^{*T}_{r(j-1)}m^{*}_{r(j-1)i}}n^{*T}_{r(j-1)}\right)m^{*}_{r(j-1)i}. \tag{94}$$

Recursively, from (91)-(94), $m'^{*}_{rji}(t)$ can be related to the known normalized Euclidean coordinate m^{*}_{r1i}. Once $m'^{*}_{rji}(t)$ is computed based on Case 2, the geometric model in Fig. 5 is equivalent to that in Fig. 2. Therefore, the Euclidean reconstruction in Section 3 can be used to build the Euclidean relationships among different coordinate frames.

4.4 Tracking and Mapping

The tracking control design is the same as that in Section 3, once the Euclidean relationship between $\mathcal{F}$ and $\mathcal{F}_d$ is obtained based on the Euclidean reconstruction analysis as shown in Section 4.3. The time-varying Euclidean position of the UGV and the stationary position of the reference objects can be localized with respect to the global coordinate system $\mathcal{I}_R$. Using the known geometric length s_{21i} and a unit normal n^{*}_{r1} (i.e., the normal to π^{*}_{1} expressed in $\mathcal{I}_R$), the geometric reconstruction method in Dupree et al. (2007); Gans et al. (2008); Mackunis et al. (2007) can be utilized to obtain $\bar{m}^{*}_{r1i}(t)$. Based on the computed $\bar{m}^{*}_{r1i}(t)$, (92) can be used to find z'^{*}_{r2i}, and then (91) can be used to find $\bar{m}'^{*}_{r2i}(t)$. Recursively, based on (93) and (94), the Euclidean coordinates of the other reference objects denoted as $\bar{m}'^{*}_{rji}(t)$ $(j = 3, ..., k)$ can be computed. Similarly, using the known geometric length s_{1i} and a unit normal $n(t)$ (i.e., the normal to π expressed in $\mathcal{I}$), the geometric reconstruction method in Gans et al. (2008) can also be utilized to obtain $\bar{m}_i(t)$. Based on the computed $\bar{m}_i(t)$, (60) and (61) can be used to find $\bar{m}'_i(t)$.

4.5 Simulation Results

A numerical simulation was performed to illustrate the localization and mapping performance given the controller in (83), (84), and the adaptive update law in (85). The origins of the coordinate frames $\mathcal{F}$, $\mathcal{F}^{*}_{1}$, $\mathcal{F}^{*}_{2}$, and $\mathcal{F}_d$, and the four coplanar feature points on the planes π, π^{*}_{1} π^{*}_{2}, and π_d are chosen such that the Euclidean coordinates of the feature points in $\mathcal{F}$, $\mathcal{F}^{*}_{1}$,

$\mathcal{F}_2^*$, and $\mathcal{F}_d$ are given by s_i (where $i = 1, 2, 3, 4$) i.e., the feature points are located at the same distance from the origins of the coordinate frames $\mathcal{F}$, $\mathcal{F}_1^*$, $\mathcal{F}_2^*$, and $\mathcal{F}_d$. The Euclidean space trajectory of the time-varying feature points attached to the plane π_d and the moving reference camera $\mathcal{I}_M$ along with the performance of the visual servo tracking controller is shown in Fig. 6, which shows the Euclidean space trajectory of the feature points attached to the planes π and π_d, taken by $\mathcal{I}$ and $\mathcal{I}_M$, respectively and the time-varying trajectory of the current and reference camera, $\mathcal{I}$ and $\mathcal{I}_M$, respectively. From Fig. 6, it can be seen that the current trajectory corresponding to the time-varying UGV $\mathcal{F}(t)$ is indistinguishable from the desired trajectory corresponding to the time-varying UGV $\mathcal{F}_d(t)$. The tracking error is depicted in Fig. 7. Fig. 8 shows the results of localization of the current UGV attached to $\mathcal{F}(t)$ and mapping of reference targets attached to $\mathcal{F}_1^*$ and $\mathcal{F}_2^*$ expressed in constant reference frame $\mathcal{I}_R$.

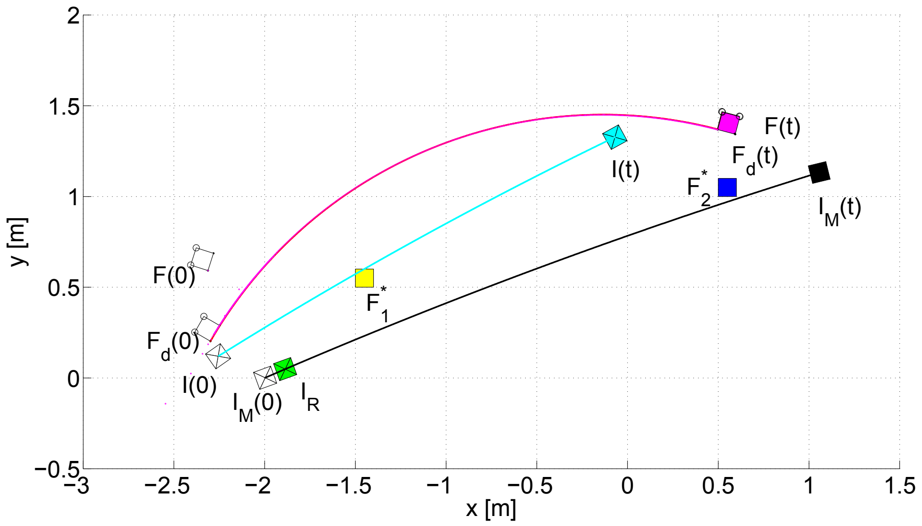


Fig. 6. Euclidean space trajectory of the feature points attached to the current (i.e. $\mathcal{F}(t)$) and desired (i.e. $\mathcal{F}_d(t)$) UGV taken by $\mathcal{I}$ and $\mathcal{I}_M$, respectively and the time-varying trajectory of the current and reference camera, $\mathcal{I}$ and $\mathcal{I}_M$, respectively. $\mathcal{F}(0)$ denotes the initial position of the current UGV, $\mathcal{F}(t)$ denotes the time-varying position of the current UGV, $\mathcal{F}_d(0)$ denotes the initial position of the desired UGV, $\mathcal{F}_d(t)$ denotes the time-varying position of the desired UGV, $\mathcal{I}(0)$ denotes the initial position of the current camera, $\mathcal{I}(t)$ denotes the time-varying position of the current camera, $\mathcal{I}_M(0)$ denotes the initial position of the time-varying reference camera, $\mathcal{I}_M(t)$ denotes the time-varying position of the time-varying reference camera, $\mathcal{I}_R$ denotes the position of the stationary reference camera, and $\mathcal{F}_1^*$ and $\mathcal{F}_2^*$ denote the position of the stationary reference objects.

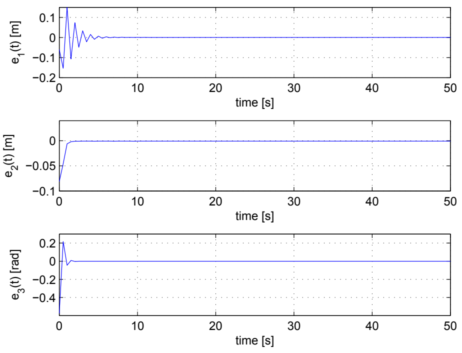


Fig. 7. Linear (i.e. $e_1(t)$ and $e_2(t)$) and angular (i.e. $e_3(t)$) tracking error.

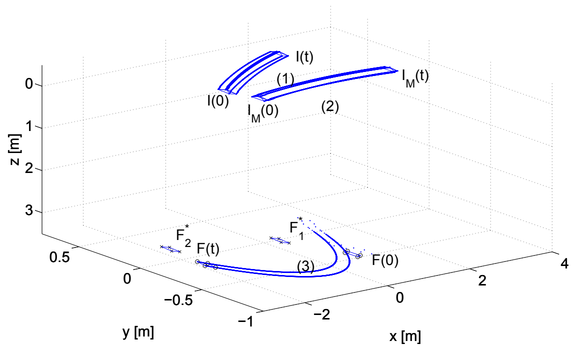


Fig. 8. Results of localization of the current UGV attached to $\mathcal{F}(t)$ and mapping of reference targets attached to $\mathcal{F}_1^*$ and $\mathcal{F}_2^*$ expressed in constant reference frame $\mathcal{I}_R$. Specifically, trajectory (1) shows the time-varying pose of the current camera attached to $\mathcal{I}(t)$, trajectory (2) shows the time-varying pose of the moving camera attached to $\mathcal{I}_M(t)$, and trajectory (3) shows the time-varying pose of the current UGV attached to $\mathcal{F}(t)$ measured in the stationary reference camera frame $\mathcal{I}_R$. $\mathcal{F}(0)$ denotes the initial position of the current UGV, $\mathcal{I}(0)$ denotes the initial position of the current camera, $\mathcal{I}_M(0)$ denotes the initial position of the time-varying reference camera, and $\mathcal{F}_1^*$ and $\mathcal{F}_2^*$ denote the position of the stationary reference objects.

5. References

Allen, P. K., Timcenko, A., Yoshimi, B. & Michelman, P. (1993). Automated tracking and grasping of a moving object with a robotic hand-eye system, *IEEE Trans. Robot. Automat.* **9**(2): 152–165.

Baker, S. & Nayar, S. (1999). A theory of single-viewpoint catadioptric image formation, *Int. J. Computer Vision* **35**(2): 175–196.

Bensalah, F. & Chaumette, F. (1995). Compensation of abrupt motion changes in target tracking by visual servoing, *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pp. 181–187.

Boufama, B. & Mohr, R. (1995). Epipole and fundamental matrix estimation using virtual parallax, *Proc. IEEE Int. Conf. Computer Vision*, pp. 1030–1036.

Burschka, D. & Hager, G. (2001). Vision-based control of mobile robots, *Proc. IEEE Int. Conf. Robot. Automat.*, pp. 1707–1713.

Chen, J., Dawson, D. M., Dixon, W. E. & Behal, A. (2005). Adaptive homography-based visual servo tracking for a fixed camera configuration with a camera-in-hand extension, *IEEE Trans. Contr. Syst. Technol.* **13**(5): 814– 825.

Chen, J., Dixon, W. E., Dawson, D. M. & McIntyre, M. (2006). Homography-based visual servo tracking control of a wheeled mobile robot, *IEEE Trans. Robot.* **22**(2): 406–415.

Das, A., Fierro, R., Kumar, V., Southall, B., Spletzer, J. & Taylor, C. (2001). Real-time vision-based control of a nonholonomic mobile robot, *Proc. IEEE Int. Conf. Robot. Automat.*, pp. 1714–1719.

Davison, A. J., Reid, I. D., Molton, N. D. & Stasse, O. (2007). MonoSLAM: Real-time single camera SLAM, *IEEE Trans. Pattern Anal. Machine Intell.* **29**(6): 1052–1067.

Dixon, W. E., Behal, A., Dawson, D. M. & Nagarkatti, S. (2003). *Nonlinear Control of Engineering Systems: A Lyapunov-Based Approach*, Birkhäuser Boston.

Dixon, W. E., Dawson, D. M., Zergeroglu, E. & Behal, A. (2001). Adaptive tracking control of a wheeled mobile robot via an uncalibrated camera system, *IEEE Trans. Syst., Man, Cybern. - Part B: Cybern.* **31**(3): 341–352.

Dupree, K., Gans, N., Mackunis, W. & Dixon, W. E. (2007). Euclidean feature tracking for a rotating satellite, *Proc. American Control Conf.*, pp. 3874–3879.

Eustice, R., Singh, H., Leonard, J., Walter, M. & Ballard, R. (2005). Visually navigating the RMS titanic with SLAM information filters, *Proc. Robotics: Science and Systems*.

Fang, Y., Dixon, W. E., Dawson, D. M. & Chawda, P. (2005). Homography-based visual servoing of wheeled mobile robots, *IEEE Trans. Syst., Man, Cybern. - Part B: Cybern.* **35**(5): 1041–1050.

Faugeras, O. & Lustman, F. (1988). Motion and structure from motion in a piecewise planar environment, *Int. J. Pattern Recognition and Artificial Intelligence* **2**(3): 485–508.

Gans, N. R., Dani, A. & Dixon, W. E. (2008). Visual servoing to an arbitrary pose with respect to an object given a single known length, *Proc. American Control Conf.*, pp. 1261–1267.

Goncalves, L., di Bernardo, E., Benson, D., Svedman, M., Ostrowski, J., Karlsson, N. & Pirjanian, P. (2005). A visual front-end for simultaneous localization and mapping, *Proc. IEEE Int. Conf. Robot. Automat.*, pp. 44–49.

Hagar, G., Kriegman, D., Georghiades, A. & Ben-Shahar, O. (1998). Toward domain-independent navigation: Dynamic vision and control,", *Proc. IEEE Conf. Decision Control*, pp. 3257–3262.

Hager, G. D., Chang, W.-C. & Morse, A. S. (1995). Robot hand-eye coordination based on stereo vision, *IEEE Contr. Syst. Mag.* **15**(1): 30–39.

Hu, G., Dixon, W. E., Gupta, S. & Fitz-coy, N. (2006). A quaternion formulation for homography-based visual servo control, *Proc. IEEE Int. Conf. Robot. Automat.*, pp. 2391–2396.

Hu, G., Gans, N., Mehta, S. & Dixon, W. E. (2007). Daisy chaining based visual servo control part II: Extensions, applications and open problems, *Proc. IEEE Multi-Conf. Syst. Control*, pp. 729–734.

Hu, G., Mehta, S., Gans, N. & Dixon, W. E. (2007). Daisy chaining based visual servo control part I: Adaptive quaternion-based tracking control, *Proc. IEEE Multi-Conf. Syst. Control*, pp. 1474–1479.

Hutchinson, S., Hager, G. & Corke, P. (1996). A tutorial on visual servo control, *IEEE Trans. Robot. Automat.* **12**(5): 651–670.

Jensfelt, P., Kragic, D., Folkesson, J. & Bjorkman, M. (2006). A framework for vision based bearing only 3D SLAM, *Proc. IEEE Int. Conf. Robot. Automat.*, pp. 1944–1950.

Jung, I.-K. & Lacroix, S. (2003). High resolution terrain mapping using low attitude aerial stereo imagery, *Proc. IEEE Int. Conf. Computer Vision*, pp. 946–951.

Kim, B., Roh, D., Lee, J., Lee, M., Son, K., Lee, M., Choi, J. & Han, S. (2001). Localization of a mobile robot using images of a moving target, *Proc. IEEE Int. Conf. Robot. Automat.*, pp. 253–258.

Kim, J.-H. & Sukkarieh, S. (2003). Airborne simultaneous localisation and map building, *Proc. IEEE Int. Conf. Robot. Automat.*, pp. 406–411.

Ma, Y., Kosecka, J. & Sastry, S. (1999). Vision guided navigation for nonholonomic mobile robot, *IEEE Trans. Robot.* **15**(3): 521–536.

Mackunis, W., Gans, N., Kaiser, K. & Dixon, W. E. (2007). Unified tracking and regulation visual servo control for wheeled mobile robots, *Proc. IEEE Multi-Conf. Syst. Control*, pp. 88–93.

Malis, E. & Chaumette, F. (2000). 2 1/2 D visual servoing with respect to unknown objects through a new estimation scheme of camera displacement, *Int. J. Computer Vision* **37**(1): 79–97.

Mehta, S., Dixon, W. E., MacArthur, D. & Crane, C. D. (2006). Visual servo control of an unmanned ground vehicle via a moving airborne monocular camera, *Proc. American Control Conf.*, pp. 5276–5211.

Mehta, S., Hu, G., Gans, N. & Dixon, W. E. (2006). Adaptive vision-based collaborative tracking control of an ugv via a moving airborne camera: A daisy chaining approach, *Proc. IEEE Conf. Decision Control*, pp. 3867–3872.

Papanikolopoulos, N., Khosla, P. & Kanade, T. (1993). Visual tracking of a moving target by a camera mounted on a robot: a combination of control and vision, *IEEE Trans. Robot. Automat.* **9**(1): 14–35.

Se, S., Lowe, D. & Little, J. (2002). Global localization using distinctive visual features, *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pp. 226–231.

Shuster, M. (1993). A survey of attitude representations, *J. Astronautical Sciences* **41**(4): 439–518.

Slotine, J. J. & Li, W. (1991). *Applied Nonlinear Control*, Prentice Hall, Inc., Englewood Cliff, NJ.

Song, K.-T. & Huang, J.-H. (2001). Fast optical flow estimation and its application to real-time obstacle avoidance, *Proc. IEEE Int. Conf. Robot. Automat.*, pp. 2891–2896.

Wiiesoma, S., Wolfe, D. & Richards, R. (1993). Eye-to-hand coordination for vision-guided robot control applications, *Int. J. Robot. Res.* **12**(1): 65–78.

Zhang, Z. & Hanson, A. R. (1995). Scaled euclidean 3D reconstruction based on externally uncalibrated cameras, *IEEE Symp. Computer Vision*, pp. 37–42.

Visual Based Localization of a Legged Robot with a topological representation

Francisco Martín[1], Vicente Matellán[2], José María Cañas[1] and Carlos Agüero[1]
[1]*Rey Juan Carlos University*
[2]*León University*
Spain

1. Introduction

One of the fundamental tasks for a mobile robot is the ability to determine its location in the environment. This ability is the basis for mobile robot operations, such as position based navigation or motion planning techniques (Borenstein et al, 1996). This problem is known as the *localization* problem.
Robots have to perform a great variety of tasks in an indoor office environment, such as delivering mail, guiding people from one room to another, and so on. In order to perform these tasks, the robot does not need precise information about its position, which could be obtained from a *geometric environment representation* approach. The approximate, but reliable, information about its position is successful.
In this work we use a markovian technique with a topological representation of the environment. This method will be conducted by a legged robot, which will be detailed in this section.
The office environment has been divided into states, in similar way to previous works as (Nourbakhsh et al, 1995) and (Kuipers, 2000) have, depending on the observations (explained later), that can be obtained in each part of this environment. So, a state could be a large portion of a corridor, for instance, where the sensory information is similar. The main advantage is the simplification of the localization process, reducing the number of states where the robot could be. On the other hand, the disadvantage is the low level of accuracy in the information about the robot's position, but this is not a problem for performing the robot tasks, as we mentioned above.
Our work has been developed using the Sony AIBO ERS7 robot (Fig. 1). This is an autonomous robot which incorporates an embedded MIPS processor running at 576MHz, and 64MB of main memory. It gets information from the environment mainly through a 350K-pixel color camera and 2 infrared sensors. AIBO's main locomotion characteristic is its canine aspect with four legs.

The main reason for choosing this robotic platform for our research is its generalization as a low cost legged robotic platform. Our group is committed to the use of common platforms and the availability of source codes, allowing research claims to be checked by peers.

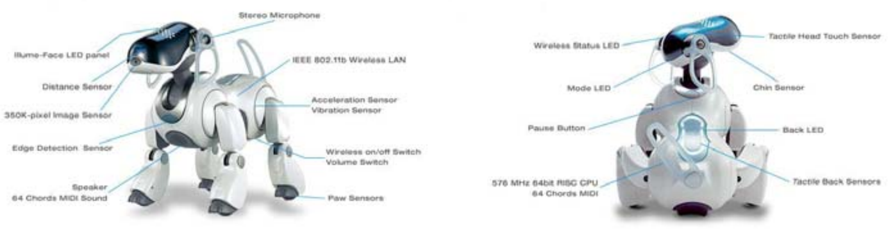


Fig. 1. Sony AIBO ERS7. (image from AIBO site)

Another key factor in this work is the limited and noisy sensory information available, obtained mainly from a camera. The legged robot locomotion influences are notable in the camera, allocated in the robot's head. This makes the image acquisition noisy. In the literature (Enderle, 2000), (Enderle, 2001), (D.Schultz & Fox, 2004) and (Veloso et al, 2000) face localization problem in legged robots using vision, but most of these works carry out their experiments in reduced and very "engineerized" environments, mainly in the Robocup four-legged league, in which landmarks are designed to be easily detected and do not present simetry or multiplicity. On the other hand, our work has been tested in a large office environment using only natural landmarks. They are harder to detect and can be similar in different locations in the environment, presenting simetry in many cases.
Most topological based localization works (e.g. (Simmons and Koening, 1995) and (Radhakrishnan and Nourbakhsh, 1999) have been developed using wheeled robots with sonar, laser sensors or omni-directional cameras. These kinds of sensors are much more reliable than cameras, and they obtain 360^{o} sensory information from the environment. These sensors and accurate odometry information make the methods used in these works inappropriate for our settings. A legged robot displacement depends greatly on the floor conditions, which makes odometry unreliable. A non-grid topological approach is used in a SLAM work, (Choset and Nagatani, 2001), where the map is a one-dimensional set of curves that captures the salient topology of the environment. Once again, this approach is unusable in our platform because of its sensor model.
The localization method proposed in this paper is based on a markovian approach (Fox et al, 1999). This is a traditional approach to solve the localization problem, however, most previous works using this approach are based on metric representation (Kosecká and Li, 2004) (where a wheeled robot is used and extracts local invariant features from images) using wheeled robots (López et al, 2003). The Markovian approach has proved to be an effective solution for mobile robot localization in large unstructured environments (Cassandra et al, 1996) (Gechter et al, 2001). This method calculates a probability density (belief) over the entire space of states (nodes of the topological map in this case).
There are other approaches that use sampling algorithms for localization. For instance, the Monte Carlo approach does not calculate a probability density over the entire set of states. This approach only calculates the localization probability of a set of samples named particles, allocated along the space of states. The amount of particles in a state determines

the robot's position. This approach has been used in applications (e.g. (Sridharan et al, 2005), (Guerrero and Solar, 2003) and (Thrun et al, 2001)) where a large number of states does not allow the use of non-sampling algorithms, mainly for scalability reasons. In our work, the space of states is small because we represent many geometric positions in the same topological state.
This paper presents the practical steps needed to make the Markovian method effective and reliable for a legged robot using its on-board camera as the main sensor over a large not-engineered indoor environment. In summary, this paper presents three ma jor contributions:

1. The topological representation of large indoor office environments in states and their use as localization representation for a legged robot.
2. The development of a probabilistic localization technique for legged robots, where odometry is not reliable or accurate.
3. The use of noisy and non omni-directional sensors (mainly cameras) for detecting natural landmarks in indoor office environments, such as doors and ceiling lights.

The remainder of this chapter is organized as follows: in section 2 we give a brief description of the Markov process developed. In section 3 we describe our model and its components, showing the experiments and results in section 4. Finally, we will put forward our conclusions in section 5.

2. The Markovian localization framework

Localization based on indirect information provided by the robot sensors (sonar, laser, etc.) has been successfully integrated into the probabilistic framework and has exhibited good results (Thrun, 2003). In particular, sampling methods that speed up the estimation (Fox et al, 1999) are currently the most popular ones (Simmons and Koning, 1995).
In our work, we have used a markovian approach in which a probability distribution Bel over all the possible locations *S = {s1 , s2 , ...}* is maintained. *Bel$_t$ (S = s)* represents the belief of being in state s at time *t*. Depending on the knowledge of the initial localization of robot *Bel$_0$ (S)*, the initial state will be uniformly distributed,

$$Bel_0(s_i) = \frac{1}{|S|} \tag{1}$$

or will be centered in a state where the initial position j is known,

$$Bel_0(s_i) = 0 + \delta, \forall i \neq j, \delta << 0.01 \tag{2}$$

$$Bel_0(s_j) = 1 - \delta, \delta << 0.01 \tag{3}$$

The belief *Bel* actualization is divided into two atomic steps. The first one is the movement step applied when an action has been performed by the robot. The belief is modified according to the action performed. The second one is the observation step, in which the belief is updated according to the observations taken from the sensors. This process is iterative and is executed every time a new movement is performed. These two steps in more detail are:

- **Movement step**. Robot actions are modelled on the probability *p(s′|s, a)* (action model). This is the probability of reaching state s′ if an action a is executed in state *s*. To obtain *the a priori* belief for the whole set of states Bel_t *(S′)*, Bayesian updating is assumed. When an action is executed we apply equation 4.

$$Bel_t(s') = \sum_{s \in S} p(s'|s, a) \cdot Bel_{t-1}(s), \forall s' \in S \tag{4}$$

- **Observation step**. To calculate the updated belief $Bel_t(S)$, we take *p(o|s)* (sensor model) as the probability of getting observation *o* to be in state s and the operation is as described in [13]. When a new independent observation is obtained, the belief is updated using equation 5.

$$Bel_t(s) = p(o_1|s) \cdot p(o_2|s) \cdots p(o_n|s) \cdot Bel_t(s'), \forall s, s' \in S \tag{5}$$

$$Bel_t(s) = p(\mathbf{o}|s) \cdot Bel_t(s'), \forall s, s' \in S \tag{6}$$

3. Our Model

Our localization method requires three components to be described, and these will be defined in more detail in this section:

1. The map and how it is translated into a set of states.
2. A set of actions that the robot can perform and their probabilistic action model related to states *p(s′|s, a)*.
3. A set of observations that the robot perceives from the environment, and its probabilistic model related to states *p(o|s)*.

3.1 The state space

We name possible locations of the robot as "states". These states are defined over an indoor office environment (see Fig. 2) made up of corridors (represented in the set of nodes as circles) and rooms (represented as boxes). Nodes are defined as places with similar characteristics and can group together one or more states. In this way, for example, a particular node could be a continuous corridor region where there are no doors to the right or left. Another node could be another continuous corridor region where there is a door to the right and a wall to the left, and so on.

In order to translate a metric map in the stated representation, first, nodes are created, according to the previous definition. Once the set of nodes has been defined, each node is divided into four different states, representing the robot's position with four orientations: North, East, South, and West.

In the top left diagram of figure 2 we can see a portion of an office environment. At the top right of the same figure, the same area has been divided into nodes attending to their characteristics. For example, a region where there are doors on both sides of the corridor is different to the region where there is only one. This division is guided by the number of doors and ceiling lights the robot can sense in each position. These natural landmarks are the ones that the robot is able to perceive more easily from its raw camera images according to our experimentation.

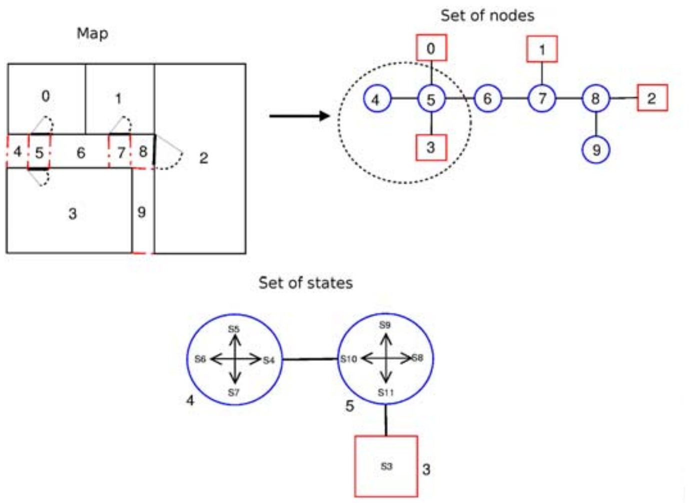


Fig. 2. The set of states is built from the map and its node decomposition

Note this is a topological organization, where nodes represent areas of different sizes. States are then defined over the nodes with different orientation. In this way, for example, in figure 2 node 4 is divided into states 4 to 7, node 5 into states 8 to 11 and so on (lower part of figure 2):

Turn Left	N: 0.15	T: 0.70	TT: 0.15	TTT: 0.0
Turn Right	N: 0.15	T: 0.70	TT: 0.15	TTT: 0.0
Go Forward	N: 0.20	F: 0.6	FF: 0.15	FFF: 0.05

Table 1. Uncertainly in action execution

3.2 The state space

The action primitives we have implemented in this work are: to turn 90º to the left $a_{\{TL\}}$, to turn 90º to the right $a_{\{TR\}}$ and go forward $a_{\{F\}}$ until the next state with the same orientation is reached. In table 1 we show the uncertain model we have used to characterize the action execution. If the robot executes “go forward”, for example, we have in mind that the robot could either do nothing(N), reach the next state (T), or go to the second state in that direction (TT) or even three states (TTT).

When the robot executes an action primitive, i.e. when the robot moves, it updates the belief as shown in (4). The action model defines $p(s'|s, a)$ as the probability of reaching state s', starting at state s and executing action a:

$$p(s'|s,a), \forall s \in S, \forall a \in A = \{a_{\{F\}}, a_{\{T_L\}}, a_{\{T_R\}}\} \tag{7}$$

This probability $p(s'|s, a)$ will represent our action model and it will be calculated *a priori*, depending on the possible action the robot can perform in that state space according to table 1. The robot's navigation is also an important aspect of this model:

- The robot must be centered in the corridor, as much as possible, in order to get correct observations from the environment.
- The robot must avoid bumping into any dynamic obstacle and, of course, walls and doors.
- Transitions from one state to the next must be detected.

We use information from camera and infrared sensors to achieve these ob jectives:

- The robot's head is moving horizontally in an arc of [-90^o ,90^o] (figure 4) at 1 Hz.
- The robot is continuously obtaining infrared information in each arc position. The robot will detect any obstacle to avoid collision. It will also use this information in order to be centred in the corridor.
- The robot will capture an image if the angle's head is near 90^o or -90^o(blue zone in figure 4). It lets us know what elements (doors or walls) there are on both sides of the robot. The robot maintains a memory of the elements that have been detected at both sides. When any of these elements change, the state will be changed too. For example, if the robot detects a door on its right and a wall on its left, it will be aware of a state change when it detects a wall on its right, for instance.

Fig. 3 shows the ground-truth path that the robot has followed along the corridor, avoiding the obstacles (persons and walls) and centering itself in it. This information has been manually recorded placing marks on the floor behind the robot at constant rate of 1 mark every 10 seconds and interpolating the path.

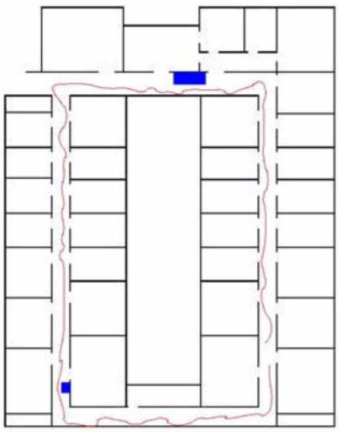


Fig. 3. Ground-truth path followed by the robot.

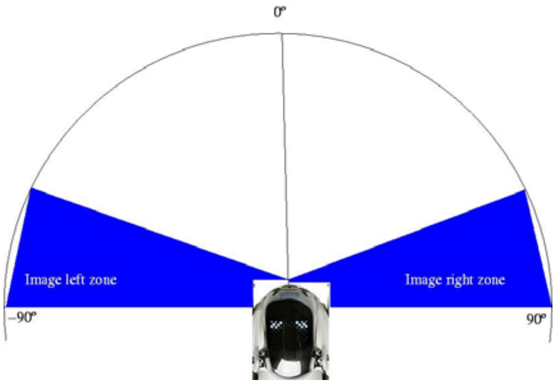


Fig. 4. Active sensing. The robot senses the whole area with its infrared sensor. It analyzes the images captured in the blue area.

3.2 Sensor Model

Our sensor model takes three types of sensations from the image obtained by the robot's camera:

- **Depth**. The main goal of this observation is to measure how far the robot is from the wall when it is oriented toward the end of the corridor. For this purpose we detect the number of ceiling lights that the robot perceives. If the number of ceiling lights is high, the robot is far from the end. If this number is low, the robot is near to the end. In figure 5 we can see the original image and the image that shows the detection of ceiling lights (green boxes).
- **Doors.** The robot is able to count the number of doors it can observe ahead using a color filter. The doors are supposed to be perpendicular to the floor and the jambs parallel to them. If a region of the image fulfills with these specifications, it is assumed to be a door. In figure 6 we can see the original image and the image with the result of the door detection (blue boxes).
- **Nearby landmarks**. This observation gives us information about which landmarks are around the robot. We define landmarks as the doors or walls that are situated on the right and left sides of the robot. For example, an observation may be that there is a door on the left side and a wall on the right side.

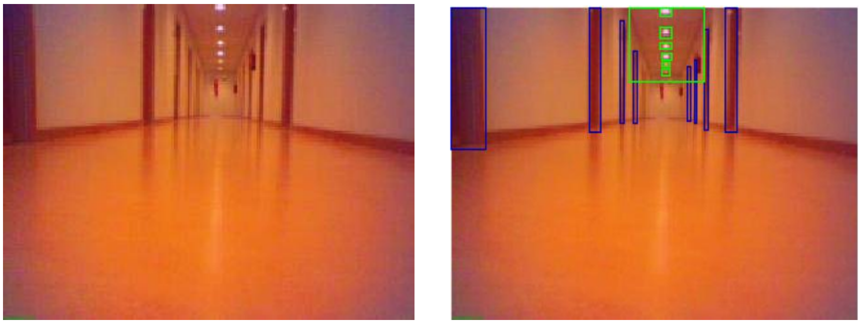


Fig. 5. Detecting 6 ceiling lights and 8 doors.

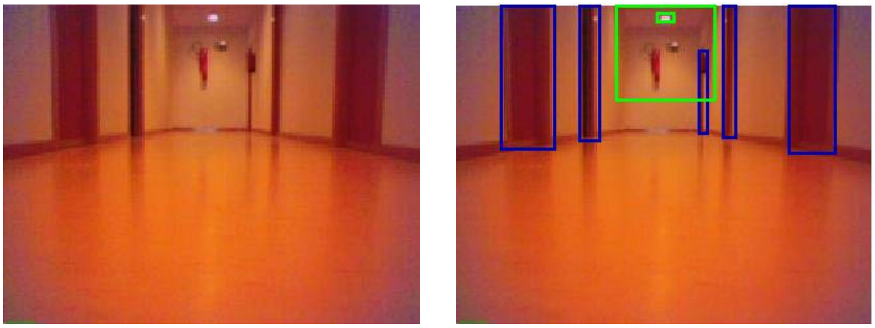


Fig. 6. Image information extraction results.

Once the data is collected, we apply equation (5) to correct the belief, according to the information obtained by the sensors. The types of sensations described before are independent from them, so we can apply equation 6.

$$Bel_{subsequent}(s) = p(\mathbf{o}|s) \cdot Bel_{previous}(s), \forall s \in S \tag{8}$$

$$\begin{aligned} Bel_{subsequent}(s) \;=\; & p(o_{ceilinglights}|s \cdot \\ & p(o_{doors}|s) \cdot \\ & p(o_{nearlandmarks}|s) \cdot \\ & Bel_{previous}(s), \forall s \in S \end{aligned} \tag{9}$$

4. Experimental setup and results

Several experiments were conducted to test the quality of the localization algorithm. The experiments were carried out in our office environment in our normal daily work. So, the environment is noisy because of the normal activity: people walking in the corridor, doors opening, and so on. This sometimes produces erroneous sensory information that will show how robust the model being used is. This is a requirement presented in section 1.
According to the desiderata presented in section 1, we evaluate the robot's localization performance based on:

- The robot's robustness to detect state changes.
- Overall accuracy.

4.2 Test for detecting state changes

In order to test the state change detection, we designed a task in which the robot was required to go forward in the corridor and to stop when a new state was detected. Fig. 7 shows an example of this test. The correct state change detections are displayed as blue circles and the wrong state change detections are displayed as red boxes. We obtained 82% accuracy in state change detections. This information has been obtained by an operator monitoring the process and verifying when the robot has correctly detected the new state.

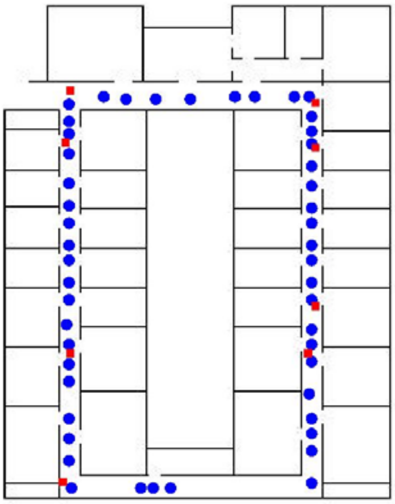


Fig. 7. Correct state change detection (blue circles) and the incorrect state change detection (red boxes).

4.2 Accuracy

To test the overall accuracy, we have designed two experiments: in the first one the initial position is not known, and in the second one the initial position is known. In each experiment we measured two values:

- **Error in localization**. To measure the error in localization, we count the nodes that a robot must traverse from the position that the localization algorithm calculates as the most probable (The mode of the *Bel* distribution), for the robot's actual position.
- **Entropy**. The entropy of the states probabilities is a usual measure for determining when the belief about the robot pose is concentrated in few states. This doesn't mean the robot is well localized. This is only a dispersion measure that indicates when the belief would determine the robot pose. When this value is near 0, the Bel probabilities are accumulated in a single node, which is considered to be the robot's position. If the entropy value is near 1, the robot's position cannot be determined and the Bel information is not useful (the error is not meaningful neither). We must consider situation in which continuous errors in perception lead robot pose converge into a wrong pose with low entropy.

$$\boldsymbol{H}(Bel) = -\sum_{s \in S} Bel(s) log(Bel(s)) \tag{10}$$

4.2.2 Initial position unknown

In this test (Figure 8), the initial position is not known. Figure 9 shows how the error and the entropy (figure 10) evolve in the localization process. At the initial step, the entropy is high, because it is distributed in several states, making it difficult to determine the robot's actual

position. In state 2, the entropy is low, which means that the belief is concentrated in few states (typically two in our case,
representing symmetry in the environment), showing instability in the error graph. When error stabilizes at step 15 and the entropy is still low, the robot has localized itself. Some localization error can happen due to observation errors or motion errors (this increases entropy), but the system recovers quickly.

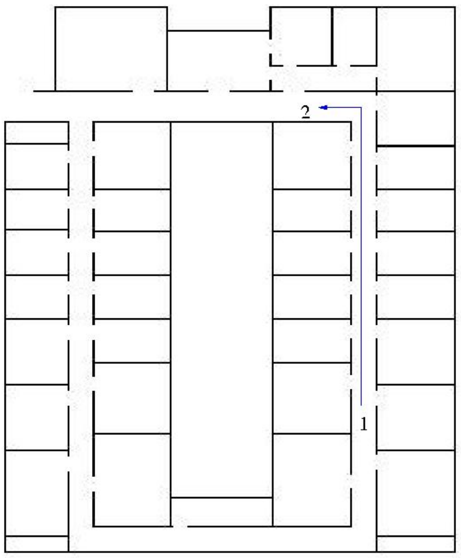


Fig. 8. The robot moves from position 1 to position 2. The robot does not know its initial position.

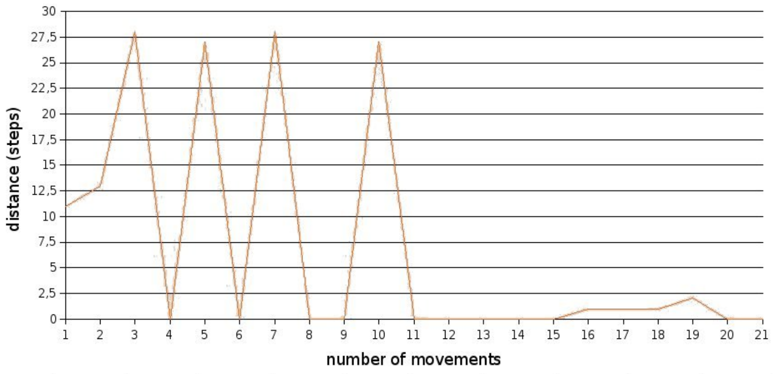


Fig. 9. Error measured as the distance from Bel mode to the actual robot position.

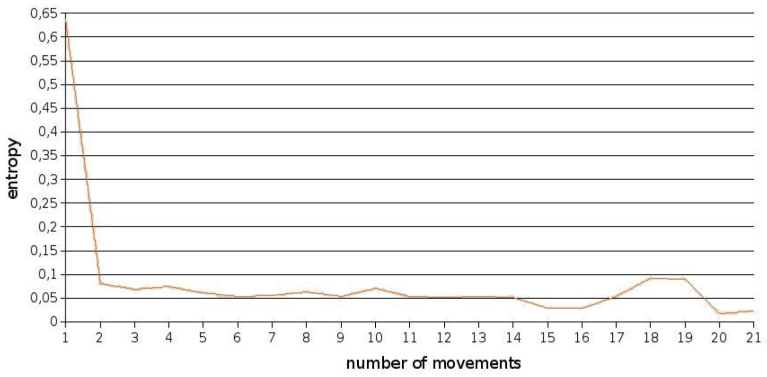


Fig. 10. *Bel* entropy

4.2.2 Initial position known

In this test (Fig. 11), the initial position is known. The robot starts at a known position and this is why the error (Fig. 12) starts at low values. Some perception errors in state 24 cause the robot to get lost for a few movements, but when these errors disappear, the robot recovers its position knowledge.

The experimentation results show interesting conclusions. First of all, our approach works and is ableto be carried out with the localization task. The experimentation results also show that the knowledge of the initial state does not influence the process in the long term.

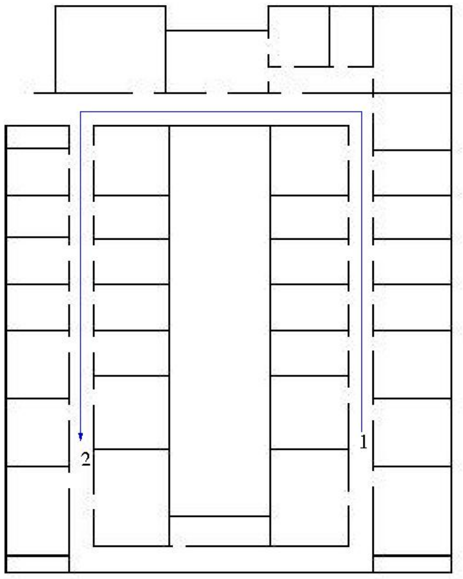


Fig. 11. The robot moves from position 1 to position 2. The robot knows its initial position.

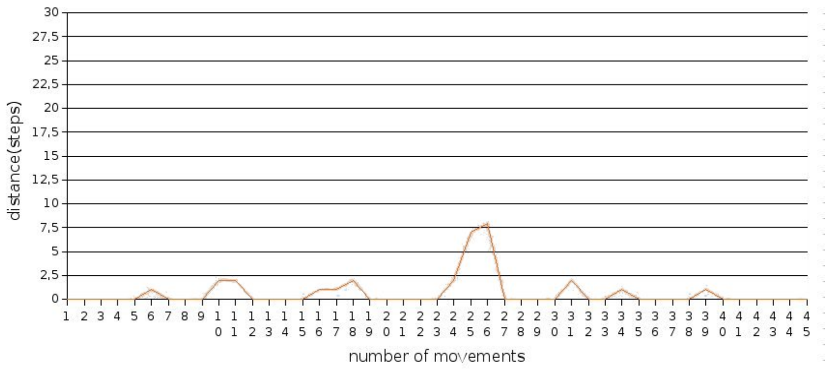


Fig. 12. Error measured as the distance from Bel mode to the actual robot position.

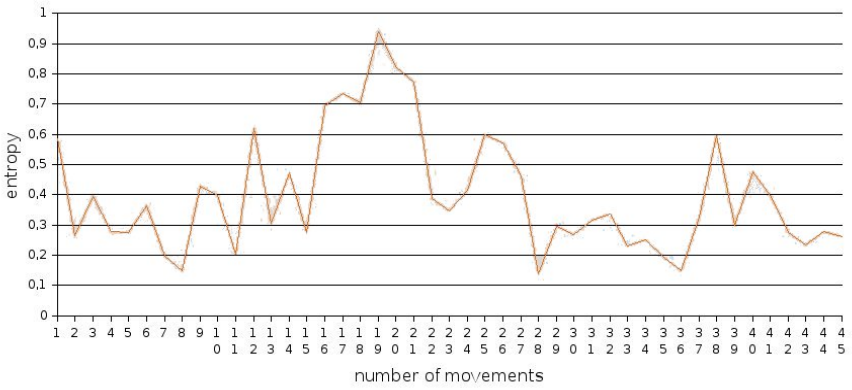


Fig. 13. *Bel* entropy.

5. Conclusions

In this chapter we have presented the performance of a localization method of legged AIBO robots in not-engineered environments, using vision as an active input sensor. This method is based on classic markovian approach but it has not been previously used with legged robots in indoor office environments. We have shown that the robot is able to localize itself in real time even inenvironments with noise produced by the human activity in a real office. It deals with uncertainty in its actions and uses perceived natural landmarks of the environment as the main sensor input.

We have also shown that data obtained from sensors, mainly the camera, is discriminate enough and allows a fast convergence from an initial unknown state, in which the belief has been distributed uniformly over the set of states . We have also shown that the robot can overcome action failures while localizing, and it recovers from them in an efficient way.

The set of observations we have chosen have been descriptive enough to be efficient in the localization process. We think that the way we determine the number of doors and ceiling lights has been the key for the success of the localization system. This approach depends on

the a priori environment knowledge, but we are studying new types of useful observations to make thisapproach applicable in other environments where this a priori knowledge is not available.
Despite these results, there are some limitations that deserve future research. One of the key limitation arises from the low accuracy in the localization due to the granularity of the large areas defined as states in the map building. Maybe granularities near to the metric approximation could be more useful for many indoor applications.
We believe that probabilistic navigation techniques hold great promise for enabling legged robots to become reliable enough to operate in real office environments.

6. References

J.Borenstein, B.Everett, and L.Feng, *Navigating mobile robots: Systems and techniques*. MA: Ltd. Wesley, 1996.

R. Nourbakhsh, R. Powers, and S. Birchfield, *Dervish - an office-navigating robot.*, AI Magazine, vol. 16, no. 2, pp. 53–60, 1995.

B. Kuipers, The spatial semantic hierarchy, Artif. Intel l., vol. 119, no. 1-2, pp. 191–233, 2000.

D.Schultz and D.Fox, *Bayesian color estimation for adaptative vision-based robot localization*, in IROS-04, Sept. 2004.

S. Enderle, M. Ritter, D. Fox, S. Sablatnog, G. K. Kraetzschmar, and G. Palm, *Soccer-Robot Localization Using Sporadic Visual Features*, in Intel ligent Autonomous Systems 6 (IAS-6) (E. Pagello, F. Groen, T. Arai, R. Dillmann, and A. Stentz, eds.), (Amsterdam, The Netherlands), pp. 959– 966, IOS Press, 2000.

S. Enderle, H. Folkerts, M. Ritter, S. Sablatnog, G. K. Kraetzschmar, and G. Palm, *Vision-Based Robot Localization Using Sporadic Features*, in Robot Vision (R. Klette, S. Peleg, and G. Sommer, eds.), vol. 1998 of Lecture Notes in Computer Science, Berlin, Heidelberg, Germany: Spinger-Verlag, 2001.

M. Veloso, E. Winner, S. Lenser, J. Bruce, and T. Balch, *Vision-servoed localization and behavior-based planning for an autonomous quadrup legged robot*, in Proceedings of the Fifth International Conference on Artificial Intel ligence Planning Systems, (Breckenridge, CO), pp. 387–394, April 2000.

R. Simmons and S. Koening, *Probabilistic navigation in partially observable environments*, in Proceedings of the 1995 International Joint Conference on Artificial Intel ligence, (Montreal (Canada)), pp. 1080–1087, July 1995.

D. Radhakrishnan and I. Nourbakhsh, *Topological localization by training a vision-based transition detector*, in Proceedings of IROS 1999, vol. 1, pp. 468 – 473, October 1999.

H. Choset and K. Nagatani, *Topological simultaneous localization and mapping (slam): toward exact localization without explicit localization*, IEEE Transactions on Robotics and Automation, vol. 17, pp. 125 – 137, April 2001.

D. Fox, W. Burgard, and S. Thrun, *Markov localization for mobile robots in dynamic environments,* Journal of Artificial Intel ligence Research, vol. 11, pp. 391–427, 1999.

J. Koseck´a and F. li, *Vision based topological markov localization*, in Proceedings of the 2004 IEEE International Conference on Robotics and Automation, (Barcelona (Spain)), Apr. 2004.

M. E. López, L. M. Bergasa, and M.S.Escudero, *Visually augmented POMDP for indoor robot navigation, Applied Informatics*, pp. 183–187, 2003.

A. R. Cassandra, L. P. Kaelbling, and J. A. Kurien, *Acting under uncertainty: Discrete bayesian models for mobile robot navigation*, in Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 1996.

F. Gechter, V. Thomas, and F. Charpillet, *Robot localization by stochastic vision based device*, in The 5th World Multi-Conference on Systemics, Cybernetics and Informatics - SCI 2001. The 7th International Conference on Information Systems Analysis and Synthesis - ISAS 2001, Orlando, FL, USA, Jul 2001.

M. Sridharan, G. Kuhlmann, and P. Stone, *Practical vision-based monte carlo localization on a legged robot*, in IEEE International Conference on Robotics and Automation, April 2005.

P. Guerrero and J. R. del Solar, *Auto-localización de un robot móvil aibo mediante el método de monte carlo*, Anales del Instituto de Ingenieros de Chile, vol. 115, no. 3, pp. 91–102, 2003.

S. Thrun, D. Fox, W. Burgard, and F. Dallaert, *Robust monte carlo localization for mobile robots*, Artif. Intel l., vol. 128, no. 1-2, pp. 99–141, 2001.

S. Thrun, *Robotic mapping: a survey*, pp. 1–35, 2003.

Mobile Robot Positioning Based on ZigBee Wireless Sensor Networks and Vision Sensor

Wang Hongbo
College of Mechanical Engineering, Yanshan University
Qinhuangdao, 066004, China

1. Introduction

In the practical application of the mobile robot, the first encountered problem is positioning. The positioning is an important basic function for an autonomous mobile robot, which is the premise of the robot to complete its mission, and is also currently the closely watched research topic (Meng et al., 2000; Wang et al., 2008; Panzieri et al., 2008). Positioning technology is intended to obtain more accurate position and orientation of the robot based on the information of the prior environment map and the information from multiple sensors. The positioning methods can be roughly categorized as relative position measurements and absolute position measurements (Zhang et al., 2005; Guccione et al., 2000). Relative positioning includes the use of encoders, gyroscopes and accelerometer (Zunaidi et al., 2006). Absolute positioning involves using beacons (active or passive) (Venet et al., 2002), global positioning systems (GPS) (Willgoss et al., 2003), landmark recognition (Borenstein et al., 1997) and model matching methods (Fang, et al., 2006).
GPS is widely used for absolute positioning of the mobile robot, but it cannot properly receive satellite signals indoors and has lower accuracy, which makes its indoor application to a mobile robot difficult (Kim and Park, 2008). In recent years, the robot positioning based on ZigBee wireless sensor networks has become a new study hotspot (Leland et al., 2006; Lai et al., 2007). ZigBee wireless sensor network is a new wireless standard having the following features: low cost, low power, low speed, short time delay, large network capacity and high reliability (Yang and Liu, 2007). Although ZigBee has higher positioning accuracy than GPS, it cannot meet the needs of accurate positioning for a mobile robot. Therefore, the more accurate poisoning method has to be employed.
Many references have approached the positioning problem of mobile robots by employing landmarks (Se et al., 2002; Armingol et al., 2002; Bais and Sablatnig, 2006). The key idea of the positioning method is to use special marks that include a wealth of geometric information under perspective projection so that the camera location can be easily computed from the image of the guide marks.
Researchers have chosen ceiling lights as landmarks since they can be easily detected due to the high contrast between the light and the ceiling surface. Also, the ceiling lights do not need to be installed specially (Wang and Ishimatsu, 2004; Nguyen et al., 2007). Therefore, we choose ceiling lights as landmarks to realize the accurate positioning of the mobile robot.

This chapter presents a positioning method for a mobile robot based on the ZigBee wireless sensor networks and vision sensor. Using the ZigBee, the absolute position of the mobile robot can be estimated. To obtain the accurate positioning, we use vision sensor (Jiang, 2008; Yang et al., 2008), and choose ceiling lights as landmarks to realize the relative positioning of the mobile robot. The method not only overcomes the shortcomings of low positioning accuracy using ZigBee wireless sensor networks, but also decreases the computation complexity of using landmark absolute positioning. To guide the mobile robot, the path planning and navigation control method are described. Finally, a mobile robot system is introduced and the positioning experiments are conducted.

2. The Positioning System Based on ZigBee

ZigBee is an emerging short-range, low-rate wireless network technology which is based on the IEEE 802.15.4. This is a standard of the IEEE wireless personal area network working group that is known as IEEE 802.15.4 technical standards.

2.1 The Composition of the ZigBee Wireless Channel

IEEE 802.15.4 works in the industrial, scientific, medical (ISM) band. It defines two working bands: the 868/915MHz and 2.4GHz frequency bands. They correspond to two physical layer standards, and both physical layers are based on the direct sequence spread spectrum (DSSS) technology. The 868MHz band is the additional ISM band in Europe and the 915MHz band is the additional ISM band in America. They cannot be used in China. In the IEEE 802.15.4, the ISM band is divided into 27 channels with three data rates. The 2.4GHz band has 16 channels of 250kb/s rate and the channel spacing is 5MHz; the 915MHz band has 10 channels of 40kb/s rate and the channel spacing is 2MHz; the 868 MHz band has only one channel of 20kb/s rate. The wireless channels used in ZigBee are shown in Fig. 1.

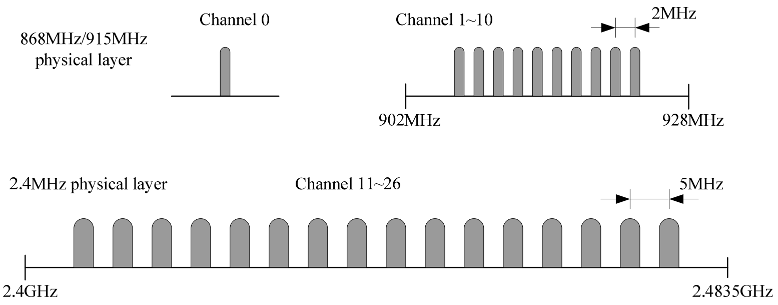


Fig. 1. The distribution of frequency and channels

2.2 Positioning System of ZigBee

The ZigBee wireless sensor networks positioning system includes two categories of nodes: the reference node and the positioning node (blind node, also known as mobile node).
A node which has a static location is called a reference node. This node must be configured with X_i and Y_i values that are corresponding to the physical location. The main task for a

reference node is to provide an information packet that contains X_i and Y_i coordinates for the blind node, also referred to as an anchor node. A reference node can be run on either a CC2430 chip or a CC2431 chip.

A blind node will communicate with the neighbouring reference nodes, collecting X_i, Y_i and RSSI (Received Signal Strength Indicator) values from each of these nodes, and calculate its position X, Y based on the parameter input using the location engine hardware. Afterwards the calculated position should be sent to a control station. This control station could be a PC or another node in the system. A blind node must use a CC2431 chip.

The biggest difference between CC2431 chip and CC2430 chip is that the CC2431 chip includes a licensed location engine hardware core from Motorola. Adopting this core, the positioning accuracy of about 0.25m that is much higher than that of GPS can be achieved, and positioning time is in the range of 40μs.

2.3 Positioning Theory

The location algorithm used in CC2431 location engine is based on RSSI values. The RSSI value will decrease when the distance increases.

The basic RSSI based ranging principle can be described as follows: The emission signal strength of the transmitting node is known, and the receiving node calculates the signal's transmission loss in spread according to the signal received. Using a theoretical or empirical model, the transmission loss will be converted into the distance.

The basis of the distance measurement based on theoretical model is the wireless signal transmission theory. In free space, the signal strength the receiver receives is given by the following formula

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L}, \tag{1}$$

where: P_t is the transmitter power; $P_r(d)$ is the receiving power; G_t and G_r are the transmitting antenna gain and receiving antenna gain, respectively; d is the distance between the sender and the receiver; L is the system loss factor; λ is the wavelength.

Based on this principle, IEEE 802.15.4 gives a simplified channel model. The received signal strength is a function of the transmitted power and the distance between the sender and the receiver. The received signal strength will decrease with increased distance and can be expressed in the following equation

$$RSSI = -(10n\log_{10} d + A), \tag{2}$$

where: n is the signal propagation constant that is related to the signal transmission environment; A is the received signal strength at a distance of one meter that is related to the transmitter power.

Figure 2 shows how the curve of the received signal strength value changes with the distance between the transmitter and the receiver.

2.4 Positioning Process

The CC2431 location engine uses distributed computing methods and hardware to compute the location. This consumes less CPU resources. The location engine collects 3 to 16 reference nodes' messages. If it receives more than 16 node's messages, it will arrange the values of d in ascending sequence and use 16 nodes of the RSSI values in front. Network communications between the adjacent nodes exchange coordinates of the position and RSSI information. In the process, the blind node obtains the RSSI values and the coordinates of adjacent reference nodes of the known address. According to the RSSI values, the blind node can calculate the distances between itself and the reference nodes. Finally, the blind node uses the Maximum Likelihood Estimation Method to calculate its coordinate.

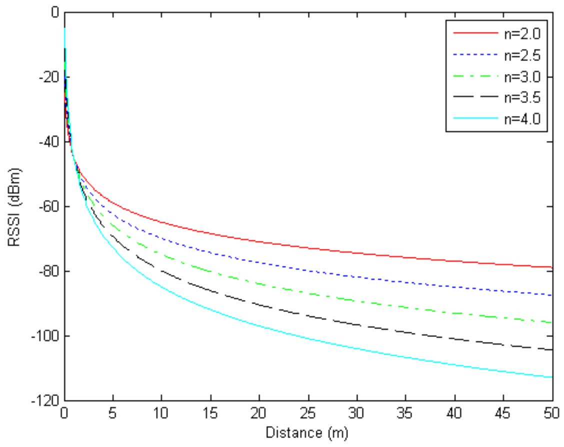


Fig. 2. The curve of the received signal strength value

The RSSI-based localization algorithm can be divided into the following steps:

(1) The blind node sends a request to all the reference nodes to obtain the signal of location information. For the blind node in the sensor networks, when it needs to determine its position, it will broadcast to all the surrounding reference nodes, requesting their coordinates and ID information.

(2) The reference nodes send their information of coordinates and ID after the blind node's request is received.

(3) The blind node receives coordinates from reference nodes with different ID, and measures RSSI values of different reference nodes.

(4) When the blind node receives the coordinates and the RSSI values of m nodes, it will convert the RSSI values to the corresponding distances d. Then it arranges the values of d in ascending sequence and establishes the corresponding collections of distances and coordinates.

Distances collection: Distance $=\{d_1, d_2, \ldots, d_m\}, d_1 < d_2 < \ldots < d_m$;

Coordinates collection: Coordinate= $\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$.

(5) In order to improve the positioning accuracy and reduce the positioning error caused by distance measured, the blind node judges the measured distances. The distance d is invalid when it is greater than the limit value δ. The distance d and the coordinate must be removed from the collections. The blind node retains the rest of the n ($n \geq 3$) group's information.

(6) For the n group's information of n reference nodes, the Maximum Likelihood Estimation Method is used to calculate the blind node's coordinate.

2.5 The Maximum Likelihood Estimation

In the three-dimensional space, if the distances from one point to 4 reference points are known, the coordinate of an unknown point can be determined. In the sensor networks, the coordinate system is two-dimensional. As long as the distances from the blind node to 3 reference nodes are known, the position of blind node can be calculated. The positioning algorithms based on ranging include Trilateration, Triangulation, Weighted Centroid Algorithm and Maximum Likelihood Estimation. In this chapter, we use the Maximum Likelihood Estimation Algorithm.

The known coordinates of n reference nodes are (x_1, y_1), (x_2, y_2), (x_3, y_3),..., (x_n, y_n), respectively. The distances between the blind node O and the n reference nodes are d_1, d_2, d_3,..., d_n. The coordinate of the blind node is (x, y). From Fig. 3, we have

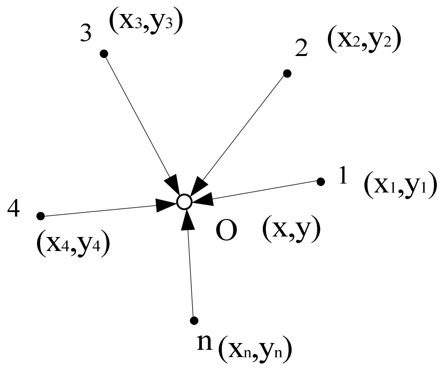


Fig. 3. Maximum Likelihood Estimation

$$\begin{cases} (x_1 - x)^2 + (y_1 - y)^2 = d_1^2 \\ \vdots \\ (x_n - x)^2 + (y_n - y)^2 = d_n^2 \end{cases}, \tag{3}$$

$$\begin{cases} x_1^2 - x_n^2 - 2(x_1 - x_n)x + y_1 - y_n^2 - 2(y_1 - y_n)y = d_1^2 - d_n^2 \\ \vdots \\ x_{n-1}^2 - x_n^2 - 2(x_{n-1} - x_n)x + y_{n-1}^2 - y_n^2 - 2(y_{n-1} - y_n)y = d_{n-1}^2 - d_n^2 \end{cases}. \tag{4}$$

The linear equations in (4) can be expressed as

$$\boldsymbol{AX} = \boldsymbol{b} \quad . \tag{5}$$

Using the Minimum Mean Square Error Method, we can obtain the coordinate of the blind node

$$\boldsymbol{X} = (\boldsymbol{A}^T \boldsymbol{A})^{-1} \boldsymbol{A}^T \boldsymbol{b}\,, \tag{6}$$

where:

$$\boldsymbol{A} = \begin{bmatrix} 2(x_1 - x_n) & 2(y_1 - y_n) \\ \vdots & \vdots \\ 2(x_{n-1} - x_n) & 2(y_{n-1} - y_n) \end{bmatrix}, \tag{7}$$

$$\boldsymbol{b} = \begin{bmatrix} x_1^2 - x_n^2 + y_1^2 - y_n^2 + d_n^2 - d_1^2 \\ \vdots \\ x_{n-1}^2 - x_n^2 + y_{n-1}^2 - y_n^2 + d_n^2 - d_{n-1}^2 \end{bmatrix}, \tag{8}$$

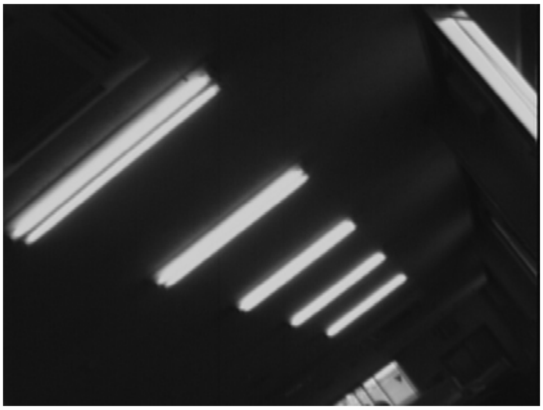
$$\boldsymbol{X} = [x \quad y]^T . \tag{9}$$

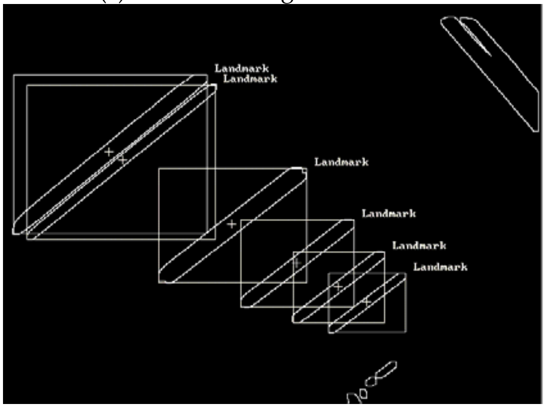
3. Positioning Based on Vision Sensor

In order to obtain the accurate positioning, the ceiling lights are chosen as landmarks and the vision sensor is used to recognize the landmarks.

3.1 Landmark Recognition

Using a vision system, the central line and contour of the ceiling light can be obtained. However, the camera may acquire some unnecessary images such as sunlight through windows or any other light sources. These unnecessary images have to be eliminated.


(a) The raw image from camera



(b) Land recognition from all light sources

Fig. 4. An example of landmark recognition

By means of the labelling program, the geometrical data (area, length, width, position, minimum and maximum horizontal/vertical coordinates) of every white cluster and the number of white clusters in the images can be obtained. From the geometrical data and the distance of the neighbouring white cluster, the desired images can be picked out from among others. An example of landmark recognition is shown in Fig. 4. Figure 4(a) shows a raw image from camera. From the labelling and geometric calculation, the ceiling lights can be selected as landmarks from all light sources as shown in Fig. 4(b).

3.2 The Coordinate System of a Single Light

On the basis of the raster coordinates of the single fluorescent light, we have the position vectors $\mathbf{p}_i = \begin{pmatrix} x_{ui} & y_{ui} & f \end{pmatrix}$ (i = 1, 2) of projection points P_i of light ends Q_i in the camera coordinate system $o_c - x_c y_c z_c$ as shown in Fig. 5. From the perspective transformation, the relation between $\mathbf{q}_i$ and $\mathbf{p}_i$ can be expressed as

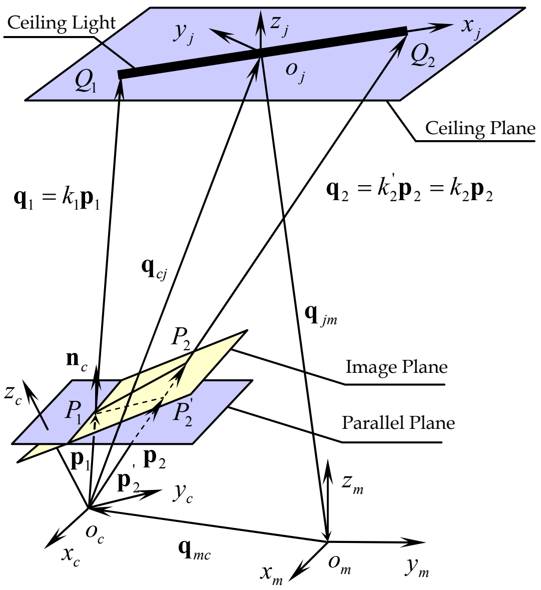


Fig. 5. Relation between the coordinate systems of the camera and a single light

$$\mathbf{q}_i = k_i \mathbf{p}_i \quad (i = 1, 2), \tag{10}$$

where $\mathbf{q}_i = \begin{pmatrix} x_{ci} & y_{ci} & z_{ci} \end{pmatrix}$ indicates the position vector of light end Q_i in the camera coordinate system, k_i is the proportionality coefficient between two vectors. If the length L_i of the light is known and the image plane and the ceiling plane are parallel, the proportionality coefficient k_i can easily be obtained as follows

$$k = k_1 = k_2 = \frac{|k_1\mathbf{p}_1|}{|\mathbf{p}_1|} = \frac{|k_2\mathbf{p}_2|}{|\mathbf{p}_2|} = \frac{|\mathbf{q}_2 - \mathbf{q}_1|}{|\mathbf{p}_2 - \mathbf{p}_1|} = \frac{L_i}{|\mathbf{p}_2 - \mathbf{p}_1|}. \tag{11}$$

Once k is known, the vector $\mathbf{q}_i$ can be obtained using equation (10).

It is more efficient to place the camera on a plane that is not parallel to the plane of the ceiling lights so that the sight of camera can be increased. In this case, equation (11) cannot be used. In order to calculate k_i, we consider a plane that passes through the point P_1 and is parallel to the ceiling plane. In the camera coordinate system, the equation of a parallel plane can be written as follows

$$\mathbf{n}_c^T(\mathbf{p}_2^{'} - \mathbf{p}_1) = 0. \tag{12}$$

In the above equation, $\mathbf{p}_2^{'}$ indicates the position vector from point o_c to point $P_2^{'}$ and the point $P_2^{'}$ is the crossing point of the parallel plane and the stright line o_cQ_2, $\mathbf{n}_c$ is the normal unit vector through the point $\mathbf{p}_1$ of the parallel plane in the camera coordinate system and can be obtained using the following equation

$$\mathbf{n}_c = \mathbf{A}^T(0 \quad 0 \quad 1)^T, \tag{13}$$

where $\mathbf{A}$ represents the rotation transformation matrix of the camera coordinate system relative to the coordinate system of mobile robot and is known in advance. The relation between the position vectors $\mathbf{p}_2^{'}$ and $\mathbf{p}_2$ can be expressed in the following equation

$$\mathbf{p}_2^{'} = t\mathbf{p}_2. \tag{14}$$

Substitute equation (14) into (12), and the parameter t can be obtained as follows

$$t = \frac{\mathbf{n}_c^T\mathbf{p}_1}{\mathbf{n}_c^T\mathbf{p}_2}. \tag{15}$$

From equation (14), the position vector $\mathbf{p}_2^{'}$ can be obtained. When the position vector $\mathbf{p}_2$ in equation (10) and (11) is replaced by $\mathbf{p}_2^{'}$, the equation (10) and (11) can be used to obtain $\mathbf{q}_i$ and k_i.

After $\mathbf{q}_i$ is obtained, the light coordinate system can be determined. The origin o_j of the light coordinate system is at the middle point of points Q_1 and Q_2. The x-axis is along the direction from Q_1 to Q_2. The unit vector of the z-axis of the light coordinate system is parallel to the z-axis of the coordinate system $o_w - x_wy_wz_w$ fixed on the mobile robot. The transformation matrix of the light coordinate system relative to the camera coordinate system can be expressed as follows

$$\mathbf{B} = \begin{bmatrix} \mathbf{e}_x & \mathbf{e}_y & \mathbf{e}_z \end{bmatrix}, \tag{16}$$

where

$$\mathbf{e}_x = \frac{\mathbf{q}_2 - \mathbf{q}_1}{|\mathbf{q}_2 - \mathbf{q}_1|}, \quad \mathbf{e}_z = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{e}_y = \mathbf{e}_z \times \mathbf{e}_x. \tag{17}$$

3.3 The Coordinate System of Double Lights

From the raster coordinates of the double lights, we have the vectors $\mathbf{p}_i = \begin{pmatrix} x_{ui} & y_{ui} & f \end{pmatrix}$ (i = 1, 2, 3, 4) in the camera coordinate system shown in Fig. 6. From perspective transformation, the relation between $\mathbf{q}_i$ and $\mathbf{p}_i$ can be expressed as

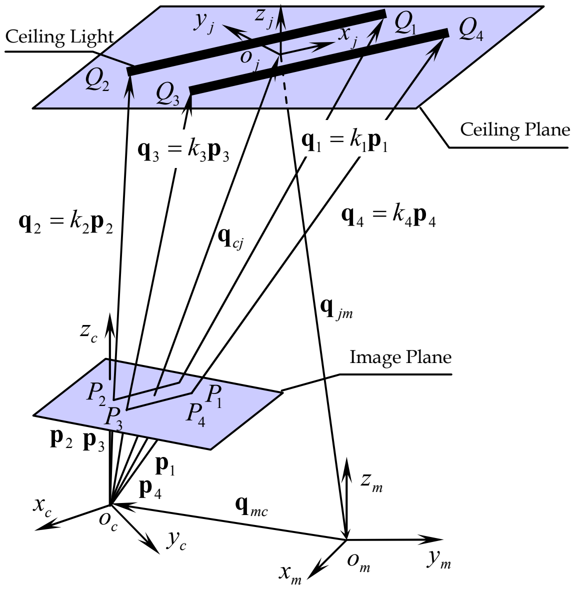


Fig.6. Relation between the coordinate systems of the camera and the double lights

$$\mathbf{q}_i = k_i\mathbf{p}_i \quad (i = 1, 2...4), \tag{18}$$

where $\mathbf{q}_i$ indicates the position vector of light end Q_i in the camera coordinate system, and k_i is the proportionality coefficient between two vectors. Since the two lights are parallel and the lengths of the two lights are equal, the following vector equation can be obtained

$$k_1\mathbf{p}_1 - k_2\mathbf{p}_2 = k_4\mathbf{p}_4 - k_3\mathbf{p}_3 \,. \tag{19}$$

From cross product of the vector $\mathbf{p}_4$ and equation (19), we have the following equation

$$k_1\left(\mathbf{p}_4 \times \mathbf{p}_1\right) - k_2\left(\mathbf{p}_4 \times \mathbf{p}_2\right) = -k_3\left(\mathbf{p}_4 \times \mathbf{p}_3\right). \tag{20}$$

By inner product of the vector $\mathbf{p}_2$ and equation (20), the following equation can be obtained

$$k_1\mathbf{p}_2 \cdot \left(\mathbf{p}_4 \times \mathbf{p}_1\right) = -k_3\mathbf{p}_2 \cdot \left(\mathbf{p}_4 \times \mathbf{p}_3\right). \tag{21}$$

The ratios of k_3 and k_1 can be expressed in the following equation

$$k_3 = c_{31}k_1 \,, c_{31} = -\frac{\mathbf{p}_2 \cdot \left(\mathbf{p}_4 \times \mathbf{p}_1\right)}{\mathbf{p}_2 \cdot \left(\mathbf{p}_4 \times \mathbf{p}_3\right)} \,. \tag{22}$$

Using a similar method, the coefficients k_2 and k_4 can be expressed as follows

$$k_4 = c_{41}k_1\,, c_{41} = \frac{\mathbf{p}_2\cdot(\mathbf{p}_3\times\mathbf{p}_1)}{\mathbf{p}_2\cdot(\mathbf{p}_3\times\mathbf{p}_4)}\,, \tag{23}$$

$$k_2 = c_{21}k_1\,, c_{21} = \frac{\mathbf{p}_3\cdot(\mathbf{p}_4\times\mathbf{p}_1)}{\mathbf{p}_3\cdot(\mathbf{p}_4\times\mathbf{p}_2)}\,. \tag{24}$$

Since the distance between Q_1 and Q_2 is the length of the lights, the following equation can be obtained

$$\left|k_2\mathbf{p}_2 - k_1\mathbf{p}_1\right| = k_1\sqrt{\mathbf{p}_1^2 - 2c_{21}(\mathbf{p}_1\cdot\mathbf{p}_2) + c_{21}^2\mathbf{p}_2^2}\,. \tag{25}$$

From the above equation, the proportionality coefficient k_1 can be calculated

$$k_1 = \frac{\left|\mathbf{q}_2 - \mathbf{q}_1\right|}{\sqrt{\mathbf{p}_1^2 - 2c_{21}(\mathbf{p}_1\cdot\mathbf{p}_2) + c_{21}^2\mathbf{p}_2^2}}\,. \tag{26}$$

The k_3, k_4 and k_2 can be obtained using equations (22), (23) and (24). From the equation (18), the $\mathbf{q}_i$ (i = 1, 2, 3, 4) can be calculated.

After $\mathbf{q}_i$ is obtained, the relation between the coordinate systems of the light and the camera can be established. The light coordinate system $o - xyz$ is set as shown in Fig. 6. The origin o of the light coordinate system is chosen at the midpoint of points Q_1 and Q_3. The position vector of the origin is $\mathbf{q}_c = (\mathbf{q}_3 + \mathbf{q}_1)/2$. The x-axis is along the direction from Q_2 to Q_1. The z-axis is along the normal line of the plane that is composed of three points Q_1, Q_2 and Q_3. The unit vector of z-axis can be determined as follows

$$\mathbf{e}_z = \frac{(\mathbf{q}_4 - \mathbf{q}_3)\times(\mathbf{q}_1 - \mathbf{q}_3)}{\left|(\mathbf{q}_4 - \mathbf{q}_3)\times(\mathbf{q}_1 - \mathbf{q}_3)\right|}\,. \tag{27}$$

The unit vector of the y-axis can be determined using right hand law. The transformation matrix $\mathbf{B}$ of the light coordinate system relative to the camera coordinate system can be obtained using equation (16).

For a light with a rectangular lampshade, the vectors $\mathbf{p}_i = \begin{pmatrix} x_{ui} & y_{ui} & f \end{pmatrix}$ (i = 1, 2, 3, 4) of the four vertices can be obtained. Using a similar method presented above, we can determine the coordinate system of the ceiling light.

3.4 Positioning of the Mobile Robot

The self-location of a mobile robot is a prerequisite step to determine the position and orientation of the mobile robot relative to the landmarks and to its world coordinate system. Since the rotation matrix $\mathbf{A}$ is known, the homogenous transformation matrix of the coordinate systems between the mobile robot and ceiling light can be expressed as

$$\mathbf{T} = \begin{bmatrix} \mathbf{A} & \mathbf{q}_{mc} \\ \mathbf{0} & 1 \end{bmatrix}\begin{bmatrix} \mathbf{B} & \mathbf{q}_{cj} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{28}$$

where $\mathbf{q}_{mc}$ indicates the position vector of the camera coordinate system relative to the mobile robot coordinate system. The inverse matrix of above matrix can be written in the following form

$$\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{C} & \mathbf{q}_{jm} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{29}$$

where

$$\mathbf{C} = \mathbf{B}^T \mathbf{A}^T , \tag{30}$$

$$\mathbf{q}_{jm} = -\mathbf{B}^T \mathbf{q}_{cj} - \mathbf{B}^T \mathbf{A}^T \mathbf{q}_{mc} . \tag{31}$$

In equation (30), **C** represents the rotation matrix of coordinate system of the mobile robot relative to ceiling light. In equation (31), $\mathbf{q}_{jm}$ expresses the position vector of the mobile robot in the coordinate system of the light. Considering that the *z*-axis of the light coordinate system and the z_m -axis of the mobile robot coordinate system are always parallel, the rotation angle α_{jm} of the mobile robot relative to the light coordinate system can be calculated as follows

$$\alpha_{jm} = \tan^{-1}\left(\frac{\sin\alpha}{\cos\alpha}\right) = \tan^{-1}\left(\frac{c_{21}}{c_{11}}\right), \tag{32}$$

where c_{21} is the element of the second row and the first column in matrix **C**.
When only one landmark is in sight of the camera, we assume the landmark is *j*-th ceiling light. Since the position vector of the *j*-th ceiling light in the world coordinate system is known, the position vector of the mobile robot in a two-dimensional world coordinate system $o-xy$ can be obtained as follows

$$\mathbf{r}_{om} = \mathbf{r}_{oj} + \mathbf{r}_{jm} , \tag{33}$$

where **r** is a two-dimensional vector with *x* and *y* coordinates as shown in Fig. 7. The orientation of the mobile robot in the world coordinate system can be obtained using the following equation

$$\alpha_{om} = \alpha_{oj} + \alpha_{jm} , \tag{34}$$

where α_{om} and α_{oj} indicate the orientation of the coordinate system fixed on the mobile robot and the *j*-th light coordinate system relative to the world coordinate system respectively.
We consider a general case where *k* landmarks are in sight of the camera as shown in Fig. 7. It is readily understandable that the more landmarks in sight are, the higher the accuracy is in the self-localization attained using the Method of Least Squares (Tajima and Komaki 1996). Since the distance between the mobile robot and landmark has an effect on the accuracy of the measurement, the weighting factor for every landmark should be considered. From equation (33), the sum of the squares of the measurement error with a weighting factor can be written as

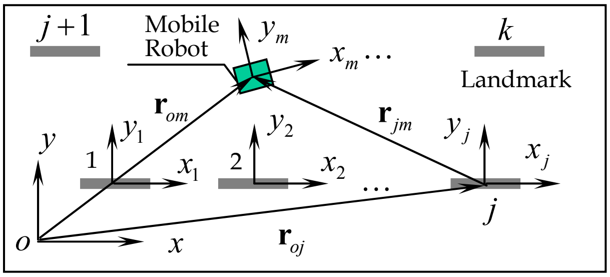


Fig. 7. Landmarks in sight of camera

$$\mathbf{J}(\mathbf{r}_{om}) = \sum_{j=1}^{k} \left(\frac{l}{l + |\mathbf{r}_{jm}|} \left(\mathbf{r}_{om} - (\mathbf{r}_{oj} + \mathbf{r}_{jm}) \right)^2 \right), \tag{35}$$

where $\frac{l}{l+|\mathbf{r}_{jm}|}$ is the weighting factor of every landmark, l can be taken as the length of the ceiling light. A necessary condition to minimize $\mathbf{J}(\mathbf{r}_{om})$ is that the following equation must be satisfied

$$\frac{\partial \mathbf{J}(\mathbf{r}_{om})}{\partial \mathbf{r}_{om}} = \sum_{j=1}^{k} \left(\frac{1}{(l + |\mathbf{r}_{jm}|)^2} \left(\mathbf{r}_{om} - (\mathbf{r}_{oj} + \mathbf{r}_{jm}) \right) \right) = 0\,. \tag{36}$$

From the above equation, the position vector $\mathbf{r}_{om}$ of the mobile robot relative to the world coordinate system can be expressed as follows

$$\mathbf{r}_{om} = \frac{\sum_{j=1}^{k} \left(\frac{1}{(l + |\mathbf{r}_{jm}|)^2} (\mathbf{r}_{oj} + \mathbf{r}_{jm}) \right)}{\sum_{j=1}^{k} \frac{1}{(l + |\mathbf{r}_{jm}|)^2}}\,. \tag{37}$$

Using a similar method, the orientation of the mobile robot relative to the world coordinate system can also be obtained as

$$\alpha_{om} = \frac{\sum_{j=1}^{k} \left(\frac{1}{(l + |\mathbf{r}_{jm}|)^2} (\alpha_{oj} + \alpha_{jm}) \right)}{\sum_{j=1}^{k} \frac{1}{(l + |\mathbf{r}_{jm}|)^2}}\,. \tag{38}$$

4. Path Planning and Navigation Control

A major problem in guiding a mobile robot to its intended goal is the path planning or find-path problem. Many references on path planning have been published to address the problem (Desaulniers and Villeneuve 2000; Zhuang et al. 2006). In this section, the method of path planning based on landmarks and the control scheme for autonomous navigation based on path planning are presented.

4.1 Navigation Map and Network Matrix

Navigation of a mobile robot using landmarks depends upon a navigation map shown in Fig. 8. The navigation map should contain the information about the position of every landmark and about possible paths. The possible paths are determined as a series of lines that connect neighboring nodes situated near the landmarks. The navigation map with *n* landmarks can be expressed using the following matrices

$$\mathbf{R}_{mark} = \begin{bmatrix} \mathbf{r}_{o1} & \mathbf{r}_{o2} & \cdots & \mathbf{r}_{on} \end{bmatrix}, \tag{39}$$

$$\Delta\mathbf{R} = \begin{bmatrix} \Delta\mathbf{r}_{11} & \Delta\mathbf{r}_{22} & \cdots & \Delta\mathbf{r}_{nn} \end{bmatrix}. \tag{40}$$

Matrix $\mathbf{R}_{mark}$ is composed of the position vectors of all landmarks (ceiling lights) in the world coordinate system. Matrix $\Delta\mathbf{R}$ indicates the position vector of every node relative to the corresponding landmark. This matrix determines the navigational course that drifts away the landmarks taking into consideration the obstacles in the path. The position vectors of all nodes can be expressed as

$$\mathbf{R}_{node} = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \cdots & \mathbf{r}_n \end{bmatrix} = \mathbf{R}_{mark} + \Delta\mathbf{R}. \tag{41}$$

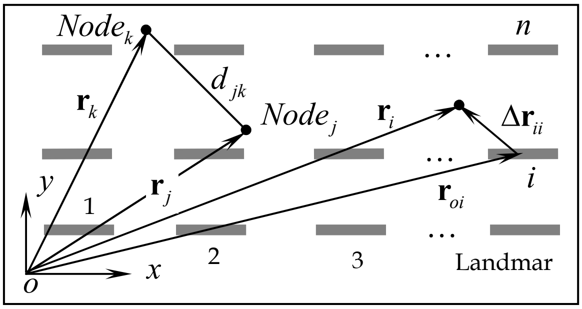


Fig. 8. Navigation map

In order to find the optimal path among all possible paths, we introduce the following network matrix

$$D^0 = \begin{bmatrix} 0 & d_{12}^0 & \cdots & d_{1n}^0 \\ d_{21}^0 & 0 & \cdots & d_{2n}^0 \\ \cdots & \cdots & \cdots & \cdots \\ d_{n1}^0 & d_{n2}^0 & \cdots & 0 \end{bmatrix}, \tag{42}$$

where d_{ij}^0 indicates the distance between node *i* and *j*, that is, $d_{ij}^0 = \left| \mathbf{r}_i - \mathbf{r}_j \right|$. If there is an obstacle between node *i* and *j*, or node *i* and *j* are not adjacent, we use $d_{ij}^0 = \infty$ to express that the path between node *i* and *j* is not feasible. Since the distance from node *i* to *j* and the distance from node *j* to *i* are equal, the network matrix is a symmetry matrix.

4.2 Shortest Path

Based on the network matrix D^0, the shortest path from the starting position to the selected node (goal position) can be found. Here, the Floyd Shortest Path Algorithm (Minieka, 1978) is used to find the shortest path. Before introducing the algorithms, some notations need to be defined:

d_{ij}^m denotes the length of a shortest path from node *i* to node *j*, where only the first *m*-1 node is allowed to be the intermediate node. If no such path exists, then let $d_{ij}^m = \infty$. From this definition of d_{ij}^m, it follows that d_{ij}^0 denotes the length of the shortest path from *i* to *j* that uses no intermediate nodes.

d_{ij}^N represents the length of a shortest path form *i* to *j*.

D^m denotes the $m \times m$ matrix whose *i*, *j*-th element is d_{ij}^m.

D^N is the matrix of shortest path length whose *i*, *j*-th element expresses the length of the shortest path among all nodes.

p_{ij}^l indicates the next-to-last node in the shortest path from *i* to *j* that is called the penultimate node of that path.

$\mathbf{V}_{path}$ represents the path vector of the shortest path whose every element is the node number (landmark number). Suppose the shortest path consists of *L* nodes, then $p_{ij}^l\,(l = 1,2,3,...L)$ constitutes the path vector $\mathbf{V}_{path} = (v_1, v_2 \cdots v_L)$ of the shortest path, where $v_l = p_{ij}^{L-l+1}$.

The Floyd Shortest Path Algorithm starts with D^0 and calculates D^1 from D^0. Next, D^2 is calculated from D^1. This process is repeated until D^N is obtained from D^{N-1}. The algorithm can be described as follows:

(1) From the navigation map, determine the network matrix D^0.

(2) For *m* = 1,2, … *N*, successively determine the elements of D^m from the elements of D^{m-1} using the following recursive formula

$$d_{ij}^m = \min\left\{d_{im}^{m-1} + d_{mj}^{m-1}, d_{ij}^{m-1}\right\}. \tag{43}$$

As each element is determined, it records the path it represents. Upon termination, the *i*, *j*-th element of matrix D^N represents the length of a shortest path from node *i* to *j*. Note that $d_{ii}^m = 0$ for all *i* and all *m*. Hence, the diagonal elements of the matrices D^1, D^2, ... D^N do not need to be calculated. Moreover, D^0 is a symmetry matrix, and only d_{ij}^m (*j* > *i*) is calculated in matrix D^m.

(3) After D^N is obtained, the penultimate node p_{ij}^l can be found as follows: Let $p_{ij}^1 = j$ and $p_{ij}^L = i$. If $p_{ij}^N = p_{ij}^0$, there are only two nodes in the shortest path and the path vector from *i* to *j* is $\mathbf{V}_{path} = (i, j) = (p_{ij}^L, p_{ij}^1)$. Otherwise,

for $p_{ij}^{l}(l = 1,2,3,...L-1)$, suppose that the penultimate node in a shortest path from *i* to *j* is any node *k*, that is, $p_{ij}^{l} = k(k \neq i, j$ and is not repeated). If *k* is satisfied with the following equation

$$d_{ik}^{N} + d_{kj}^{0} = d_{ij}^{N} \ , \tag{44}$$

we have $p_{ij}^{2} = k$. Then, the second-to-last node in this path is the penultimate node p_{ik}^{3} on a shortest path from *i* to *k*. This process can be repeated until all the nodes in this path from *i* to *j* have been tracked back. In this step, only D^{0} and D^{N} matrices are required for the value of *k* to be determined.

(4) When all the nodes in the shortest path from *i* to *j* have been found, the path vector from *i* to *j* can be obtained as follows

$$\mathbf{V}_{path} = \left(v_1, v_2 \cdots v_L\right) = \left(p_{ij}^{L}, p_{ij}^{L-1} \cdots p_{ij}^{1}\right). \tag{45}$$

4.3 Autonomous Navigation Based on Path Planning

In this section, we will present the control scheme for the autonomous navigation of the mobile robot based on the path vector. When the path vector is known, the position vectors of all nodes in the shortest path can be expressed in the following matrix

$$\mathbf{R}_{v} = \left[\mathbf{r}_{v1} \quad \mathbf{r}_{v2} \quad \ldots \quad \mathbf{r}_{vL}\right]. \tag{46}$$

From Fig. 9, two vectors adjoining three nodes in the path can be expressed as

$$\mathbf{a}_{i} = \mathbf{r}_{vi+1} - \mathbf{r}_{vi} \ , \mathbf{a}_{i+1} = \mathbf{r}_{vi+2} - \mathbf{r}_{vi+1} \quad (i \leq L\text{-}2) . \tag{47}$$

The angle between the two vectors can be obtained using the following equation

$$\alpha = \tan^{-1}\left(\frac{\left(\mathbf{a}_{i+1} \times \mathbf{a}_{i}\right) \cdot \mathbf{k}}{\mathbf{a}_{i+1} \cdot \mathbf{a}_{i}}\right), \tag{48}$$

where **k** represents the normal unit vector of the plane *o-xy*. Using the above equation, we can obtain the control scheme for the mobile robot from node *i*+1 to node *i*+2. When α < 0, the mobile robot turns α degree to the right, otherwise, it turns α degree to the left.

In the above control scheme, the mobile robot must navigate along the vector $\mathbf{a}_{i}$ and the moving direction must be coincident with the direction of the vector $\mathbf{a}_{i}$. However, when a mobile robot navigates itself from node *i* to node *i*+1, it will unavoidably encounter an obstacle. After the obstacle is avoided, the mobile robot will drift off the vector $\mathbf{a}_{i}$ as shown in Fig. 10. In order to enable the mobile robot to move to node *i*+1 after obstacle avoidance, the angle between two vectors $\mathbf{b}_{i}$ and $\mathbf{c}_{i}$ should be obtained

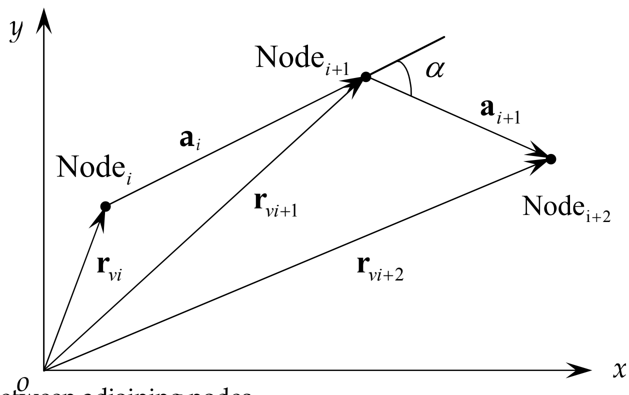


Fig. 9. Vector between adjoining nodes

$$\beta = \tan^{-1}\left(\frac{(\mathbf{c}_i \times \mathbf{b}_i)\cdot\mathbf{k}}{\mathbf{c}_i \cdot \mathbf{b}_i}\right), \tag{49}$$

where $\mathbf{b}_i = \mathbf{r}_{vi+1} - \mathbf{r}_{om}$. When the mobile robot moves to node i+1, the control scheme for the mobile robot from node i+1 to node i+2 can be obtained as follows

$$\gamma = \tan^{-1}\left(\frac{(\mathbf{a}_{i+1} \times \mathbf{b}_i)\cdot\mathbf{k}}{\mathbf{a}_{i+1} \cdot \mathbf{b}_i}\right). \tag{50}$$

When the mobile robot navigates from node i to i+1, it needs to know whether the node i+1 has been reached. The position vector between the mobile robot and node i can be obtained as follows

$$\mathbf{d}_i = \mathbf{r}_{om} - \mathbf{r}_{vi}. \tag{51}$$

The angle between the vectors $\mathbf{a}_i$ and $\mathbf{d}_i$ is calculated in the following equation

$$\theta = \tan^{-1}\left(\frac{(\mathbf{d}_i \times \mathbf{a}_i)\cdot\mathbf{k}}{\mathbf{d}_i \cdot \mathbf{a}_i}\right). \tag{52}$$

From Fig. 10, the projection of vectors $\mathbf{d}_i$ and $\mathbf{b}_i$ on the vector $\mathbf{a}_i$ can be calculated as follows

$$L_a = |\mathbf{d}_i|\cos\theta + |\mathbf{b}_i|\cos(\gamma - \alpha). \tag{53}$$

When the distance between two nodes is bigger than the projection of the vectors $\mathbf{b}_i$ and $\mathbf{d}_i$ on vector $\mathbf{a}_i$, that is, $|\mathbf{a}_i| > L_a$, the mobile robot has not yet reached the node i+1. When $|\mathbf{a}_i| \le L_a$ is obtained, node i+1 has been arrived at. In this case, the mobile robot can automatically turn to the right or to the left at angle γ and navigate from node i+1 to i+2.

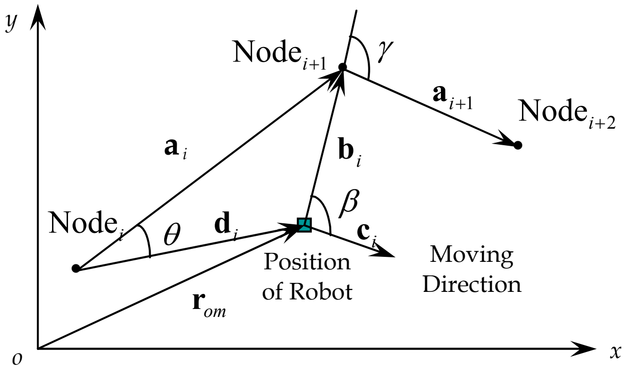


Fig. 10. Position and moving direction of the mobile robot after obstacle avoidance

5. Mobile Robot System and Positioning Experiments

5.1 Mobile Robot System

Figure 11 shows an omni-directional mobile robot developed by us. The mobile robot has 8 wheels as shown in Fig. 12. The 8 wheels are divided into 4 groups that have the same transmission mechanism. One wheel of each group is the driving wheel and the other is the free wheel. Since two belts driven by two motors make four driving wheels move in a synchronous way, the motion of the robot platform is a translation motion. The mechanical design and kinematics of the mobile robot were discussed in earlier work (Wang, et al., 2008).

The control system of the mobile robot consists of one set of the ZigBee wireless sensor networks (C51RF-3-ZDK), a CCD camera (BASLER A301FC), a laser sensor (BANNER LT3), 8 ultrasonic sensors (BTE054: US Sensor 2), a motor controller (PI-16), three all-in-one motors, a SH7044 microcomputer and a notebook PC as shown in Fig. 13. The ZigBee wireless sensor networks and the vision sensor are employed for the robot positioning, and the ultrasonic sensors are used to detect obstacles in the way. The laser sensor is not used in this experiment.

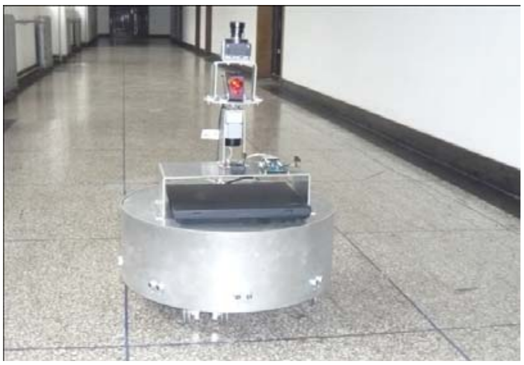


Fig. 11. The omni-directional mobile robot

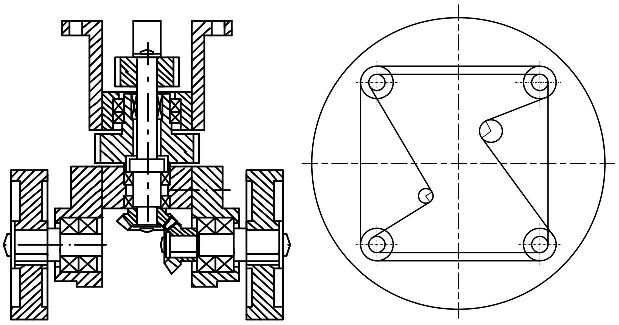


Fig. 12. The driving system of the mobile robot

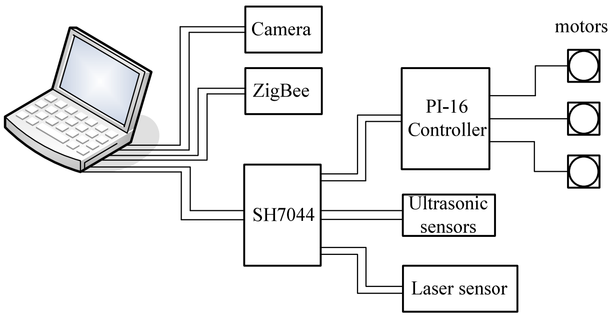


Fig. 13. The control system of the mobile robot

5.2 Positioning Experiments

In this paper, the positioning system of ZigBee wireless sensor networks includes 1 blind node and 8 reference nodes. The experiments were carried out in the corridor of our laboratory (Fig. 11). The blind node is fixed on the mobile robot, 8 reference nodes are installed in the navigational environment as shown in Fig. 14.

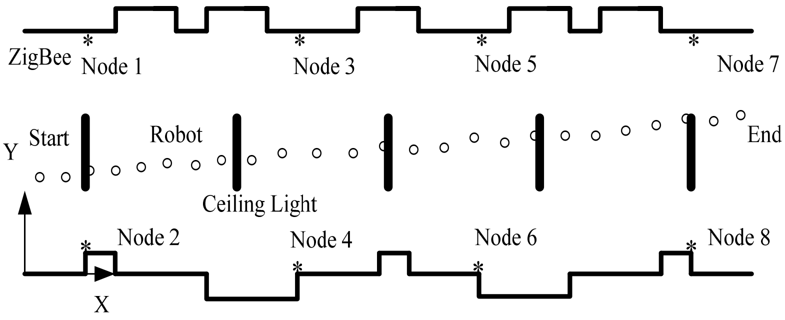


Fig. 14. Experimental environment

We let the robot navigate from the start position to the end position, and collected the positioning data. We obtained the errors of positioning by comparing the experimental data with the theoretical calculation results. The position errors are shown in Fig. 15.

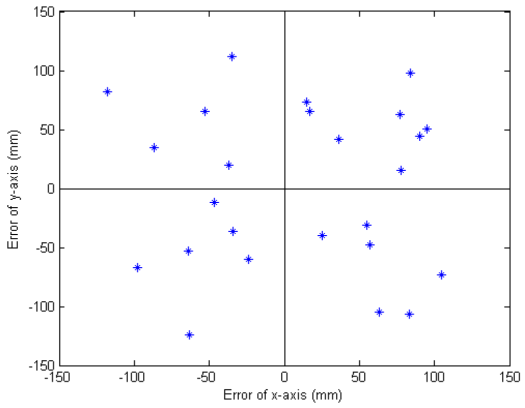


Fig. 15. Position errors

6. Conclusion

A positioning method of the mobile robot based on the ZigBee wireless sensor networks and vision sensor was presented. Using the ZigBee positioning system, the rough absolute positioning could be obtained. The positioning system and positioning theory of ZigBee were described. In order to obtain the accurate positioning, the ceiling lights were chosen as landmarks and the vision sensor was used to recognize the landmarks. The positioning method using the ceiling light landmarks was proposed. The navigation map that contains the information about the position of every landmark and possible path in the navigational environment was established and the network matrix that expressed the navigation map was obtained. Then the control method of autonomous navigation was described. The positioning experiments based on the ZigBee and vision sensor for the mobile robot developed by our laboratory were conducted and experimental results showed the effectiveness of the positioning method.

Acknowledgement

This work was supported by the Foundation of Hebei Education Committee under Grant 2008149, and the Scientific and Technological Foundation for the Returned Overseas Chinese Scholars, Hebei Province.

7. References

Armingol, J.M., Escalera, A., Moreno, L. and Salichs, M.A. (2002). Mobile robot localization using a non-linear evolutionary filter. *Advanced Robotics*, Vol. 16, No. 7, pp.629-652. ISSN: 0169-1864, 1568-5535(Online).

Bais, A. and Sablatnig, R. (2006). Landmark based global self-localization of mobile soccer robots. *Lecture Notes in Computer Science,* Springer Berlin, Heidelberg, Vol. 3852, pp.842-851. ISBN: 978-3-540-31244-4. ISSN: 0302-9743, 1611-3349 (Online).

Borenstein, J., Everett, H.R., Feng, L. and Wehe, D. (1997). Mobile robot positioning: sensors and techniques. *Journal of Robotic Systems*, Vol. 14, No. 4, pp.231-249. ISSN: 0741-2223.

Desaulniers, G. and Villeneuve, D. (2000). The shortest path problem with time windows and linear waiting costs. *Transportation Science*, Vol. 34, No. 3. pp. 312-319. ISSN: 1526-5447.

Fang F., Ma, X.D. and Dai, X.Z. (2006). Mobile robot localization based on improved model matching in hough space. *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Beijing, China. pp. 1541-1546. ISBN: 1-4244-0259-X.

Guccione, S., Muscato, G., Nunnari, G., et al. (2000). Robots for volcanos: the state of the art. *Proceedings of the 3rd International Conference on Climbing and Walking Robots*, Madrid, Spain, pp.777-788. ISSN: 1860582680.

Jiang, Z.H. (2008). Vision-based cartesian space motion control for flexible robotic manipulators. *Int. Journal of Modelling, Identification and Control*, Vol. 4, No. 4, pp.406-414. ISSN: 1746-6172.

Kim, S.Y. and Park, S. (2008). Estimation of absolute positioning of mobile robot using U-SAT. *Lecture Notes in Computer Science*, Springer Berlin, Heidelberg, Vol. 4672, pp. 143-150. ISBN: 978-3-540-74783-3. ISSN: 0302-9743, 1611-3349 (Online).

Lai, X.Z., Yang, S.X., Zeng, G.X., She, J.H. and Wu, M. (2007). New distributed positioning algorithm based on centroid of circular belt for wireless sensor networks. *Int. Journal of Automation and Computing*, Vol. 4, No. 3, pp.315-324. ISSN: 1476-8186.

Leland, E., Bradford, K. and Jenkins, O.C. (2006). Robot localization and control. *Circuit Cellar*, Issue 188, pp. 36-39. ISSN: 1528-0608.

Meng, Q.H., Sun, Y.C. and Cao, Z.L. (2000). Adaptive extended Kalman filter (AEKF) -based mobile robot localization using sonar. *Robotica*, Vol. 18, pp. 459-473. ISSN: 0263-5747.

Minieka, E. (1978). *Optimization algorithms for networks and graphs*. Marcel Dekker Inc., New York. ISBN: 0824766423.

Nguyen, V.T., Jeong, M.S., Ahn, S.M., et al., (2007). A robust localization method for mobile robots based on ceiling landmarks. *Lecture Notes in Computer Science*, Springer-Verlag, Vol. 4617, pp.422-430. ISBN: 978-3-540-73728-5. ISSN: 0302-9743, 1611-3349 (Online).

Panzieri, S., Pascucci, F. and Setola, R. (2008). Simultaneous localization and mapping of a mobile robot via interlaced extended Kalman filter. *International Journal of Modelling, Identification & Contro*l, Vol. 4, No. 1, pp.68-78. ISSN: 1746-6172.

Se, S., Lowe, D. and Little, J. (2002). Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *International Journal of Robotics Research*, Vol. 21, No. 8, pp.735-758. ISSN: 0278-3649.

Venet, T., Capitaine, T., Hamzaoui, M. and Fazzino, F. (2002). One active beacon for an indoor absolute localization of a mobile vehicle. *IEEE International Conference on Robotics and Automation*, Vol. 1, pp.1-6. ISBN: 0-7803-7272-7.

Wang, H.B. and Ishimatsu, T. (2005). Vision-based navigation for an electric wheelchair using ceiling light landmark. *Journal of Intelligent and Robotic Systems*, Vol. 41, pp.283-314. ISSN: 0921-0296.

Wang, H.B., Tian, X.B., Zhang, H.M. and Huang, Z. (2008). Application of different mechanism in omni-directional mobile robot. *Machine Design and Research*, Vol. 2, pp.25-26. ISSN: 1006-2343.

Wang, H.M., Hou, Z.G., Cheng, L. and Tan, M. (2008). Online mapping with a mobile robot in dynamic and unknown environments. *International Journal of Modelling, Identification and Control*, Vol. 4, No. 4, pp.415-423. ISSN: 1746-6172.

Willgoss, R., Rosenfeld, V., Billingsley, J. (2003). High precision GPS guidance of mobile robots. *Proceedings of the Australian Conference on Robotics and Automation,* Brisbane, Australia. pp.1-6. ISBN: 0-9587583-5-2.

Yang, P., Wu, W.Y., Moniri, M. and Chibelushi, C.C. (2008). A sensor-based SLAM algorithm for camera tracking in virtual studio. *Int. Journal of Automation and Computing*, Vol. 5, No. 2, pp.152-162. ISSN: 1476-8186, 1751-8520 (Online).

Yang, X.P. and Liu, S.Y. (2007). Mobile robot locating and tracking system design based on wireless sensor network. *Chinese Journal of Electron Devices*, Vol. 30, No. 6, pp.2265-2268. ISSN: 1005-9490.

Zhuang, H.Z., Du, S.X. and Wu, T.J. (2006). On-line real-time path planning of mobile robots in dynamic uncertain environment. *Journal of Zhejiang University: Science*. Vol. 7, No. 4, pp. 516-524. ISSN: 1673-565X.

Zhang, Y., Luo, Y. and Wang, J.F. (2005). A novel method of range measuring for a mobile robot based on multi-sensor information fusion. *Journal of Physics*: *Conference Series 13*, pp.127-132. ISSN: 1742-6588, 1742-6596 (Online).

Zunaidi, I., Kato, N., Nomura, Y. and Matsui, H. (2006). Positioning system for 4-wheel mobile robot: encoder, gyro and accelerometer data fusion with error model method. *CMU Journal*, Vol. 5, No. 1, pp.1-14. ISSN: 1685-1994.

A WSNs-based Approach and System for Mobile Robot Navigation

Huawei Liang, Tao Mei and Max Q.-H. Meng
Institute of Intelligent Machine, Chinese Academy of Sciences
China

1. Introduction

Mobile Robots are expected to do some routine or danger tasks automatically for human in many situations. Among the techniques related to the Mobile Robot, localization and navigation are the core techniques for Mobile Robots to realize true intelligence and complete autonomous moving.
J. J. Leonard and H. F. Durrant-Whyte summarized the problem of navigation into answering the following three questions (Leonard & Durrant-Whyte, 1991) : "where am I? ". "where am I going?" and "how should I get there". The first question lies to identifying the current location of the robot. The second and third questions are related to the capability of environment perceiving and path planning.
The navigation methods that are frequently used in Mobile Robots mainly include inertial navigation, visual navigation, sensor-based navigation and satellite navigation. Among the satellite navigation systems that are in use, the Global Positioning System (GPS) (Bock & Leppard, 1990) gives the most accurate information. But in some cases, the satellite system cannot or should not be used. To solve this problem, Wireless Sensor Networks (WSNs) was introduced into this area. In this chapter, a new navigation approach is proposed based on the localization function of Wireless Sensor Networks as a supplement to current navigation methods. The WSN can obtain various types of information about the environment such as the temperature, the humidity and the slope of the ground, and the proposed approach then use them to help the decision-making in navigation and path planning.
The chapter mainly includes 3 sections. The first section is about map building. A WSN-based method for environment modelling and map building was proposed. This method utilized the distributed environment information obtained by the WSN to establish the environment model and the grid map with multiple attributes. Simulative analysis showed that the environment model established by this method achieved good match result on the map. In section 2, the dynamic monitoring function of WSN was utilized to adapt the Mobile Robot to the requirement of navigation on the changing environment. An on-line path planning method based on WSNs was proposed. Using this planning method, the Mobile Robot could make trade-off between safety and efficiency through adjusting the parameters used in the algorithm. At last, an experimental WSN-based Mobile Robot

navigation system was designed and implemented to verifying the proposed navigation approach. The experimental result showed that the proposed approach could fit the application requirement of Mobile Robot navigation.

2. A WSNs-based Map Building Method for Mobile Robots

2.1 Introduction

Navigation is the key technology for mobile robot performing various tasks in a complex environment while mapping the surrounding environment and then planning the path on the basis of this map are core problems for autonomous move of the mobile robot. Map building has long been focused by researchers in corresponding fields. The process of mapping is actually a course for the mobile robot to perceive information in the surrounding environment. With the characters of quick deployments, convenience in being hidden and high quality in fault tolerance, WSN is especially fit for real-time information acquisition in unknown dynamic environment and therefore the prospects of WSN applied in mapping for mobile robot is extensive.

In a known environment, an artificial map of it is generally used for navigation as prior knowledge; in an environment without prior knowledge, the method of simultaneous localization and mapping (SLAM) are widely employed by researchers, through which on one hand they can mapping the working field of the mobile robot and on the other hand, they can localize the mobile robot (Davison & Kita, 2001) (Wang & Thorpe, 2002) (Sim & Roy, 2005); Sometimes it is relatively easier for a swarm of robots to perform a particular task than for a single robot to perform it. On the basis of this principle, multi-robot distribute map building is presented, in which the information about the local environment acquired by the robot itself are integrated with the information acquired by other robots to generate a global map. The typical feature of this method is its higher efficiency (Fenwick et al., 2002) (Burgard et al., 2000) (Yamauchi, 1999). The constructed map requires a proper representation to meet the need of path planning. There are three generally used methods for map representation: topological map, geometric feature map and gird-based map. Topological map builds a map from a global view of the studied environment; geometric feature map builds a map from the set of objects in the environment; grid-based map builds a map referring the details in the environment and it thereby is a comparably accurate method. Carnegie Mellon University (CMU) takes this kind of map representation method for its mars rover (Yahja, 1998).

In this section, the map building method based on WSNs are studied. To begin with, massive WSN nodes are deployed in the unknown environment, and then those WSN nodes transfer the distributable information acquired by them to the mobile robot. With this information, the studied environment is constructed and then the gird-based maps of this environment are also been constructed. By this way, after WSN nodes are deployed the global map can be quickly obtained, with which a global path planning can be performed to avoid possible blindness in local path planning.

2.2 Method description

The presented method can be described in the following five steps:
1. To deploy WSN nodes into the home range of the mobile robot.
2. The coordinates and sensor data of WSN nodes are transferred from WSN nodes to mobile robot. In the studied environment of our project, excessive high of the temperature, the humidity and the gradient will all obstruct the movement of the mobile robot and the factors of temperature, humidity and slope gradient are taken into account. However, the presented method is not limited to these three factors and other factors can be considered in the similar way according to the real situation.
3. With the sensor information of every WSN node, we can rebuild the estimated environment. The estimated environment is expressed in the form of temperature potential, humidity potential and gradient potential.
4. The estimated information in the environment can be binarized based on the constructed environment and predefined threshold and subsequently a grid-based map corresponding to some specific information.

Whether or not there is an obstacle in the map depends on the pre-defined threshold. In practice, the threshold is determined by the real property of the mobile robot. If the temperature at certain position is higher than the threshold of temperature, the position is thought to serve as an obsolete. Thus, a temperature map is obtained. Other maps such as humidity map and gradient map also can be gained in the similar way. As expressed in the equation (1).

$$f_{(x,y)} = \begin{cases} 1 & if\, g_{(x,y)} \geq g_0 \\ 0 & else \end{cases} \tag{1}$$

Where g(x,y) is the estimated sensor information in the point of (x, y) ; g0 is the threshold; f(x,y) is the obstacle value in the point of (x, y) . f(x,y) is a Boolean variable which is 1 when obstructed and is 0 when not obstructed.

5. To fuse the data of the map provided in step 4 and subsequently gets a synthetic grid-based map integrating multiple information.

A grid is obstructed when at least one of these information exceeds the corresponding predefined threshold. Consequently, a final map created by multi-information is acquired. The fusion equation from single information to multi-information is described in equation (2).

$$f_{overall(x,y)} = f_{temperature(x,y)} \mid f_{gradient(x,y)} \mid f_{humidity(x,y)} \tag{2}$$

Where $\mathrm{f}_{\mathrm{temperatur\ e(x, y)}}$, $\mathrm{f}_{\mathrm{gradient(x\ ,y)}}$, $\mathrm{f}_{\mathrm{humidity(x\ ,y)}}$ are the obstacle value of temperature, gradient and humidity respectively. $\mathrm{f}_{\mathrm{overall(x,y)}}$ is the obstacle value of multi-information." | " is the logical operation of or.

2.3 Simulation

Extensive simulations are performed to weigh the effectivity of this presented method. We studied a square area of 100m×100m, in which there are 250 (the number is determined according to the need our project) WSN nodes, in each of which a temperature sensor, a humidity sensor and a gradient sensor are equipped to acquire corresponding signals.

A. Construction of Simulation Environment

Firstly, we model the situations of temperature, humidity and gradient in the active environment of the mobile robot. The temperature differences in this environment result from some source of heat distributing in this area (fig.2); the humidity differences in this environment due to lakes and marshes in this area. Where there is a lake, there is a highest humidity; where there is a marsh, there is a higher humidity; where neither a lake nor a marsh is located, the humidity there is the lowest (fig.3);The gradient differences in this environment result from geological factors (fig.4). For other specific environment, similar method is also accessible through taking other factors obstructing the mobile robot into consideration.

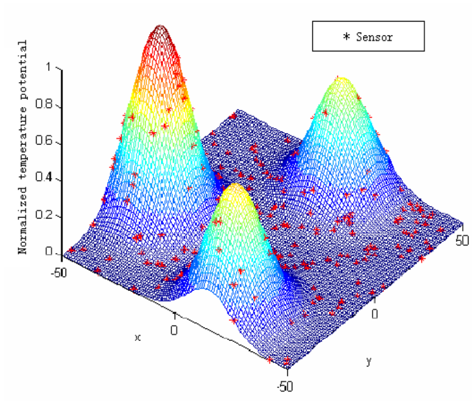


Fig. 1. Normalized temperature potential

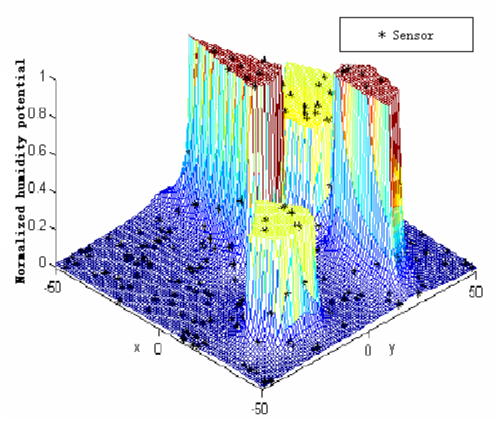


Fig. 2. Normalized humidity potential

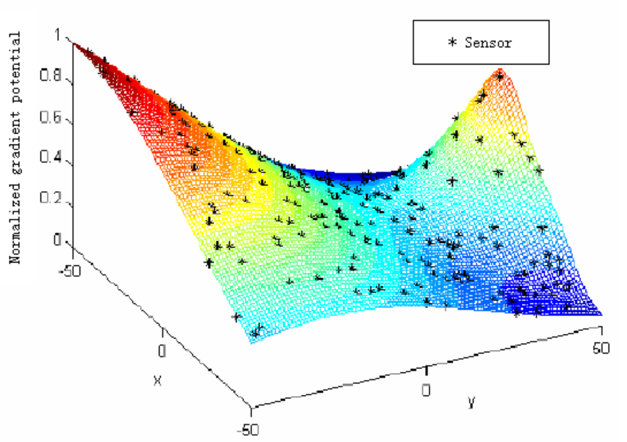


Fig. 3. Normalized gradient potential

B. Construction of the Environment Based on WSNs

In the monitoring area, after deploying 250 WSN nodes, the studied area is divided into many triangular sub regions with WSN nodes as their vertexes applying the method of Delaunay triangulation (fig.4) .

Implement interpolation for temperature, humidity and gradient information according to equation (3), the performance after interpolation is shown in fig.5, fig6, and fig7.

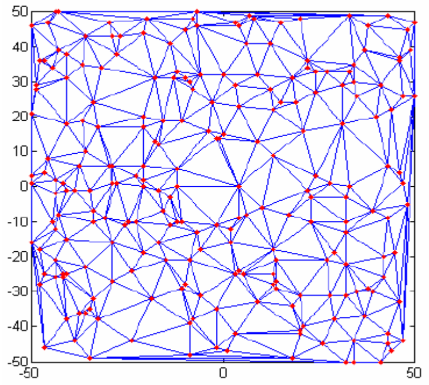


Fig. 4. Delaunay triangulation

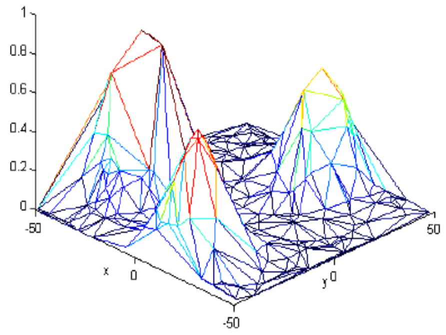


Fig. 5. Temperature potential after construction

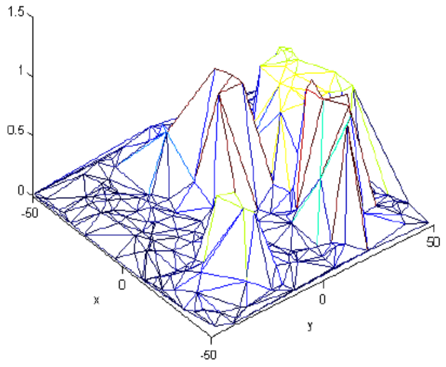


Fig. 6. Humidity potential after construction

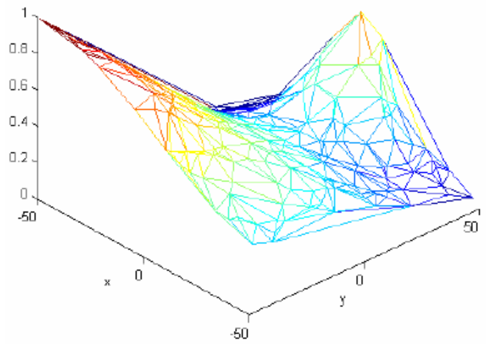


Fig. 7. Gradient potential after construction

C. Create Grid-based Map with the Constructed Environment

In simulation, the normalized temperature threshold, humidity threshold and gradient threshold are respectively designate as 0.35, 0.65 and 0.75 according to the practical situation. Thus, when the temperature in certain point is higher than 0.35, the point is thought to be obstructed by an obstacle and otherwise not be obstructed. Humidity and gradient are processed in the similar way. The grid-based maps of slope gradient, temperature and humidity obtained from the original ideal environment are respectively shown in fig.8 (1), (2), (3) while the grid-based maps of gradient, temperature and humidity obtained from the previous constructed environment are respectively shown in fig.9 (1), (2), (3), in which we can see the map error is relatively large near the edges of this area because some regions near edges are not located in any angular patch formed by WSN nodes, about which we only known little and the mobile robot thereby should avoid to pass through these regions. Consequently, on the basis of the consideration above, these regions are designated as obstacles artificially. Multi-information maps are built basing on single information maps as shown in fig.8 (4) and in fig.9 (4).

D. Error Analysis of Map Building

For a real system, the localization module of the WSN nodes has errors and there is measuring error for the equipped sensor in the node and the presented method based on limited WSN nodes also serve to introduce errors. All these errors contribute to bring about a synthetic error for the constructed multi-information map. Here we take the final error into account through extensive simulation.

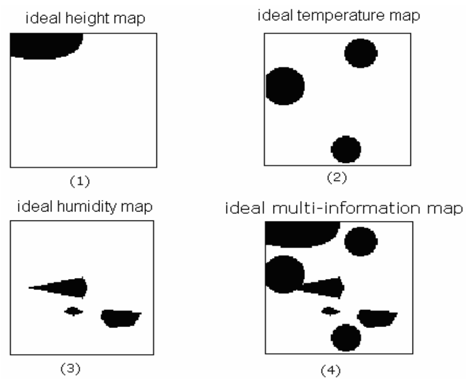


Fig. 8. Ideal map

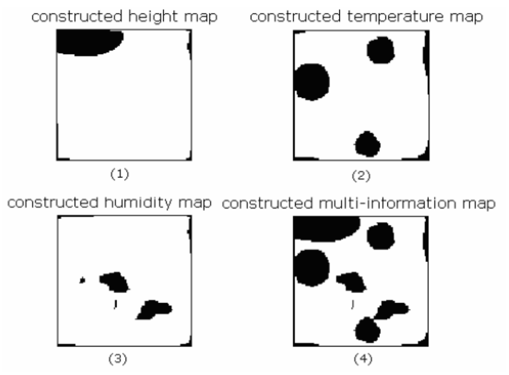

constructed height map
constructed temperature map
(1)
(2)
constructed humidity map
constructed multi-information map
(3)
(4)

Fig. 9. Constructed map

We evaluate the effectivity of this presented method by comparing differences in the Boolean value between the constructed multi-information map and the ideal multi-information map. The error is calculated by the number of grids of which the logical value of the constructed multi-information map is different from that of the ideal multi-information map. The ratio of this error and the number of all grids in the map is defined as relative error. In simulation, the final results are computed by averaging each result of ten times.

WSN nodes number	Mapping error（%）					mean（%）	Standard deviation (%)
	1	2	3	4	5		
250	11.59	13.33	12.21	12.10	11.39	12.12	0.76
500	8.90	9.25	8.94	9.97	9.65	9.34	0.46
1000	3.77	4.29	4.00	4.44	3.82	4.064	0.29

Table 1. Mapping error analysis

The ranging error of the localization module is set to be 5% and the sensor errors of gradient, temperature and humidity are also set to be 5%. On this basis, three set of results are acquired through three simulations in which 250,500 and 1000 WSN nodes are respectively deployed. The simulation result is shown in table 1, from which we can see that the final error is not very large only by deploying a relatively small quantity of WSN nodes and the more WSN nodes are deployed, the smaller the error and the standard deviation of the final result and when a large amount of WSN nodes are deployed, the final error is very small. With the booming development in radio frequency (RF), integrate circuit (IC) and micro electromechanically system (MEMS), WSN nodes are becoming much smaller in size and much cheaper in price and the intensive deployment thereby are becoming more accessible which will bring about the essential reduction in final error of the presented method.

2.4 Conclusion

In this section, a map building method for mobile robot based on WSNs is presented. Combining the features of WSNs, a concrete operational scheme is also presented, in which several factors affecting the traffic ability characteristic of the mobile robot are taking into consideration enhancing the practicality of the map for subsequent navigation. What is more, the final map is expressed in the form of general used grid-based map. Extensive simulation indicates that the error of this method satisfy general needs for navigation and for situation requiring a relatively high accuracy, increasing the amount of WSN nodes is feasible to reduce the final error to some degree.

3. A WSNs-based Path Planning Method for Mobile Robot

3.1 Introduction

The main task of path planning for the mobile robot is searching an optimal path according the environmental model where the mobile robot safely arrives at the destinations without any collision.Currently, there are many methods about path planning for the mobile robot. According to whether environmental information is known or not, the mobile robot path planning algorithms can be divided into global path planning and local path planning.
Global path planning is based on priori information of the global environment. It uses the static global map for path planning. Because the global environment information is known, it can be used to optimize some performance indicators. However, the static information in the global path planning can not work well in dynamic environment. The local path planning, can obtain the environmental information in real time, such as location, shape and size information of the obstacles, through the sensors. The local path planning more suitable for the applications in dynamic environment.
The local path planning methods mainly include: Artificial Potential Field Algorithm, Genetic Algorithm and so on. Artificial Potential Field (Khatib, 1986) has the advantage of simpleness so that it is easy to be implemented. However, it has the local optimal solution problem, and also may cause dead lock (Yoram, et al. 1991). Genetic Algorithm (Holland, 1973) is more likely to obtain the global optimal solution, while it needs more memory space and operation time.
A wireless sensor network is consisted of a large amount of sensor nodes deployed densely in the monitoring environment. Through the positioning, temperature, vibration, light, electromagnetic and radiation sensors, we can obtain a wide range of global environmental information and use them to build a omnibearing map of global environment for mobile robots. Furthermore, wireless sensor networks can achieve real-time information of the environment, base on which the robot can make on-line path planning in the dynamic environment.
In this section, we propose an on-line path planning method for mobile robot which uses a wireless sensor network to obtain the dynamic environment information.

3.2 Method description

Here, we introduce an on-line path planning method based on WSNs. Artificial potential field algorithm is used in this method, while the WSN provides a real time information of the dynamic environment. The sensor nodes are used as landmarks during robot navigation. After arriving at a landmark, the robot choose next landmark to move toward according to the destination and navigation cost. The robot repeats this process until it reaches the destination.
Navigation costs are related to the potential of the sensor nodes and the distance between sensor nodes and destination. Higher potential means larger cost, while shorter distance induce lower cost. The mobile robot always chooses the landmark with the lower cost and makes itself safely arrive at the destination. The potential of a sensor node is related to the environment information of it, such as temperature , gradient and so on.
A. Set up the cost
Sensor nodes can obtain their own locations using self-localization mechanism, while several kinds of sensors can be ultilized to acquire different kinds of environment information and then translated into the potential.

$$U^m = \begin{cases} 0 & if(D_s \le D_{tl}) \\ \dfrac{D_s - D_{tl}}{D_{th} - D_{tl}} & if(D_{tl} < D_s < D_{th}) \\ 1 & if(D_{th} < D_s) \end{cases} \tag{3}$$

$$U_n = \max(U_n^1, U_n^2, \dots) \tag{4}$$

In equation (3), Ds is the sampled environmental information, Dth and Dtl denotes the upper and lower threshold, while Um denotes the potential of some environment factor. Equation (4) is used to integrate multi-environmental information, it takes the maximum potential among all of them.

We suppose the potential values will decrease with reciprocal of the square of the distance. So neighbor sensor node s which has a distance d to node n , sensor n will have the potential value:

$$U_{ns} = kU_s \tag{5}$$

Us is the potential of node s ; k is the coefficient of attenuation , so we can obtain this equation.

$$k = \begin{cases} 1 & \text{if } d \le 1 \\ \dfrac{1}{d^2} & \text{else} \end{cases} \tag{6}$$

We take the maximum remaining potential value among them from one hop to sensor node n as the effect of the neighbor sensor nodes , so the environmental potential value $U_n^{'}$ of sensor node n:

$$U_n^{'} = \frac{U_n + U_{ns}}{1+k} \tag{7}$$

Suppose coordinate of node P in the environment is (xp, yp)， the potential value of point P is Up，the destination is T(xt, yt)，so the cost Cp from sensor P to the destination is：

$$C_p = \begin{cases} \infty & \text{if}(U_p = 1) \\ \sqrt{(x_p - x_t)^2 + (y_p - y_t)^2} + aU_p & \text{else} \end{cases} \tag{8}$$

After the sensor nodes in the working environment for mobile robot obtain the destination address, they can calculate their cost of the navigation from themselves to the destination. The mobile robot always choose the node with lowest cost as the next landmark.

B. Path selection strategy

Since the potential data and position data are obtained by the sensors, in general, the measured data surround the sensors is the most accurate and becoming worse with increasing the distance from the sensors to other places. In other words, the information surround the sensors are relatively reliable. Therefore, in the moving process, the mobile robot should walk along with the connection between sensor nodes and try the best to avoid deviating from the sensor nodes.

Under the above principles, we suppose at the moment the mobile robot places in location of the sensor node P and the corresponding path selection process is as follows:
(1) The robot set node P as arrived sign;
(2) The robot communications with P-node to obtain all the navigation cost of the 1-hop neighbor nodes of the P-node
(3) select the least cost node N as the next goal which has not arrived;
(4) If the node P does not exist non-infinite navigation price of the next node, the robot from the node P get back to the previous node and re- select the least navigation price of sensor node as the next goal node;
(5) The robot moves along with the connection path from the current node to the next node;
(6) Repeat these steps until the robot arrives at the destination.
In the moving process, when the robot has reached some node that will be set with arrived sigh which denotes that the sensor node has been reached and never reached again. Thus, we can effectively avoid falling into the loop. If there is no existing finite cost of navigation in the next step nodes, the robot will get back to the previous node and reselect the navigation path. So, we can avoid the oscillation and the loop through these methods.

3.3 Simulation

To verify the performance of the proposed method, we conducted simulation to study the method.
As shown in Fig. 10and Fig. 11, there are slope and temperature distribution models in the working space of mobile robot. The working area is 100m × 100m. navigation task is to make the mobile robot move from (10,10) to (90, 90) without any collision.
Sensor nodes are randomly deployed in the working space of mobile robot. Every node is equipped with temperature sensor and inclinometer. The communication radius among the nodes is 10m. As shown in Fig. 12 and Fig. 13, there are sampling results of slope and temperature information. The sampling error is set to 5%.

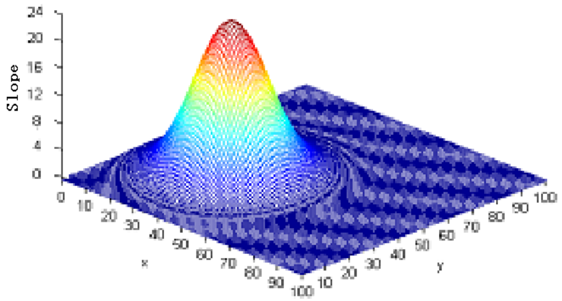


Fig. 10. Slop environment

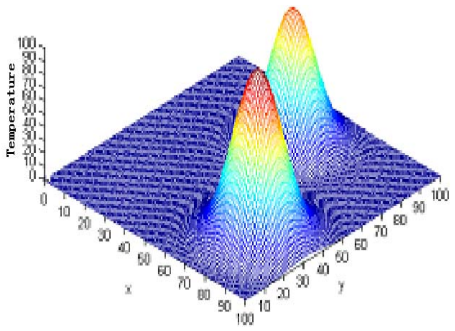


Fig. 11. Temperature environment

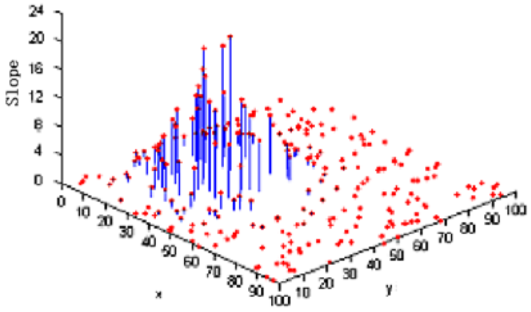


Fig. 12 Sampling Result of Slope Information

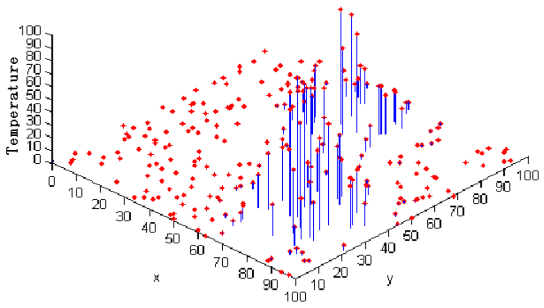


Fig. 13. Sampling Result of Temperature Information

We suppose that the upper threshold of temperature Dth is 50℃ and lower threshold Dtl is 20℃; The upper threshold of slope Dth is 10 and the lower threshold Dtl is 1. We will process Fig. 12 and Fig. 13 according to equation (3) to (4) to obtain their standard and integration results.

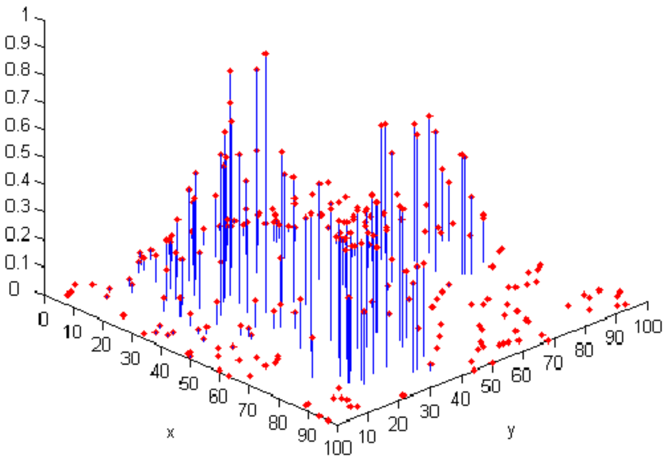


Fig. 14. Distribution of environment potential

Before the navigation, the mobile robot will firstly broadcast the destination address and “a” value (the parameter in equation (8)) to sensor nodes in the working space and every node will calculate the cost of navigation according to their potential value, destination address and “a”.

The cost of navigation is the basis of path planning for mobile robot. The robot will choose the low cost of navigation sensor node to walk gradually toward the destination. In the process of navigation, robot can choose a relatively short path or a relatively safe path through adjusting the “a” parameter value in the equation (8). Fig. 15 and Fig. 16 gives the path planning result in the cases of a parameter equates to 40 and 20.

In the process of the navigation, the robot does not need to obtain the distribution of Environmental Potential and distribution of cost of navigation. In fact, the robot choose next step objective only on the basis of the cost of navigation of the 1-hop neighbor nodes.

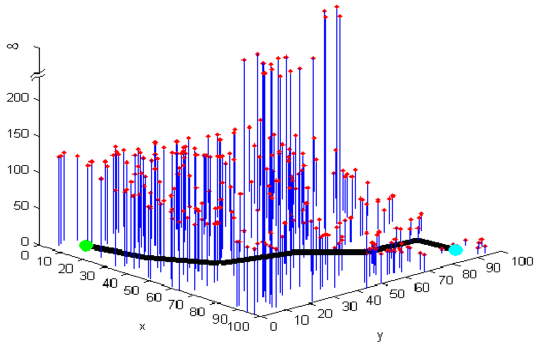


Fig. 15. Path planning result when a=40

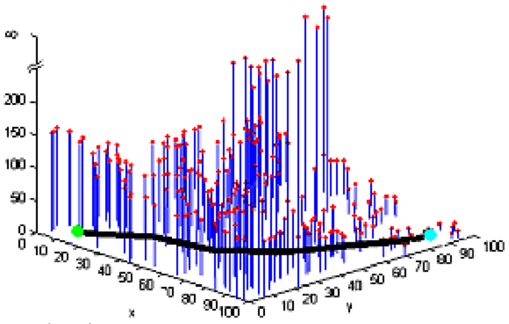


Fig. 16. Path planning result when a=20

According to the Fig. 10 and Fig. 11, we obtain gray environment model(Fig. 17) through normalizing and integrating the information. Then, using the threshold shown before, we continue to binarize the information to get the robot model(Fig. 18).

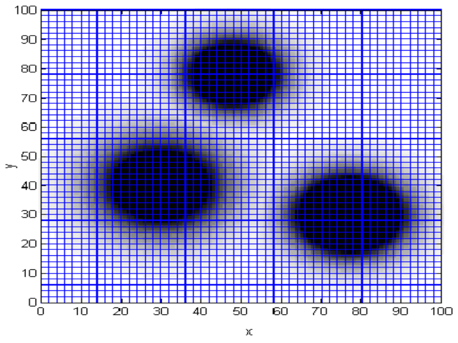


Fig. 17. Gray Environmental Model

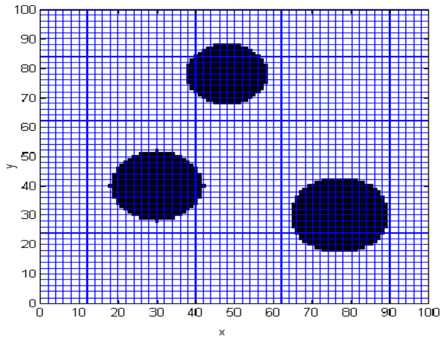


Fig. 18. Robot Mode

In order to know the effect of the path, we put the path into the constructed environment map to analyze.

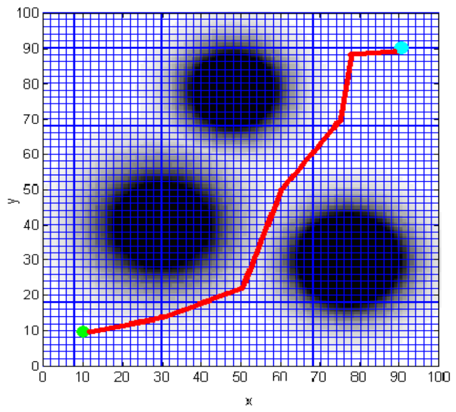


Fig. 19. Path planning result when a = 40

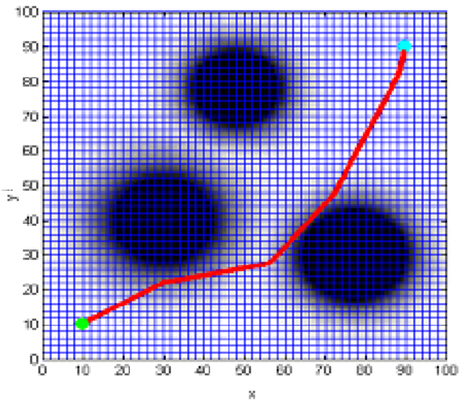


Fig. 20. Path planning result when a=20

It can be seen that when a=40, the robot chooses the safe path that can completely avoid the obstructions; when a=20, the robot chooses the shorter path although there are some obstacles in the path. However, the obstacles of the region are not beyond the limited ability of the robot so that the robot can also pass through them on the cost of paying a larger price.
In sum, we can achieve the on-line navigation for mobile robot based on the proposed method. Through adjusting the "a" parameter in the cost of navigation, the robot can get trade-off between safety and effectiveness.
This on-line path planning method based on wireless sensor networks has many advantages such as easily implemented, environmental adaptability, and low configuration requirements so that it is suitable for the battlefield environment, disaster relief and many other applications.

4. Implement of WSN-based Mobile Robot Navigation System

4.1 Introduction

The sections above mainly introduces the map building method and path planning method for mobile robot navigation based on wireless sensor networks. In this section, we develop a WSN-based navigation system using the technologies described above. Several experiments are carried out to study the validity of the system.

The mobile robot navigation system based on wireless sensor networks is mainly consist of a mobile robot, a base controller and the wireless sensor network. Wireless sensor network is composed of a large number of sensor nodes, which are scattered in the working area of robot. Sensor node is equipped with slope sensor and temperature sensor. The measurement accuracy of the slope sensor is 1◦. The sensor nodes also have a locating function based on the RSSI algorithm.

An AS-R robot is used as the mobile robot, which has the fastest mobile speed of 2.5m/s. On the other hand, a laptop is used as the base controller, which sends the commands and teleoperates the robot. The commands are relayed by the wireless sensor network. Fig. 10 shows the structure of the whole system.

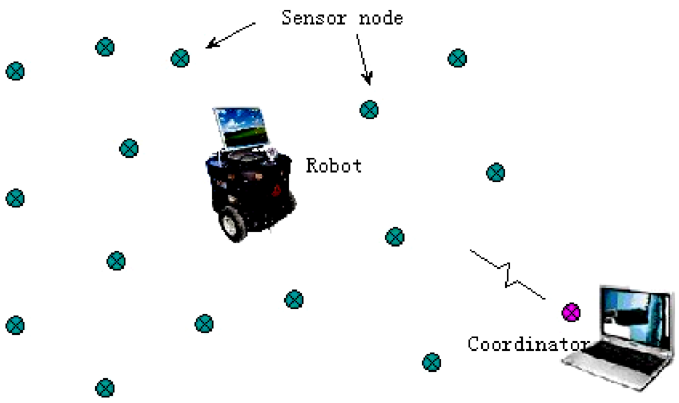


Fig. 21. The system structure

As shown in Fig. 21, the AS-R robot carries a sensor node, which we call mobile node, to locate itself. On the other hand, the base is also equipped with a sensor node, which we call coordinator node, to manage the WSN. The mobile node sends its location information to the coordinator in real time through the WSN, while the base obtains the robot location information from the coordinator, and displays it on GUI. Also, using the GUI of the base, control commands can be sent to the robot to assign tasks. The environment information obtained by WSN is transmitted to AS-R so that it can carriy out the mission of map building and path planning.

4.2 The experiments

Several experiments are carried out to study the performance of our system. A 20m x 20m square field is used as the working area of robot. Take the southwest corner of the field as the origin, divide the field into 2m x 2m cells. 2m is used as the unit in the experiments.

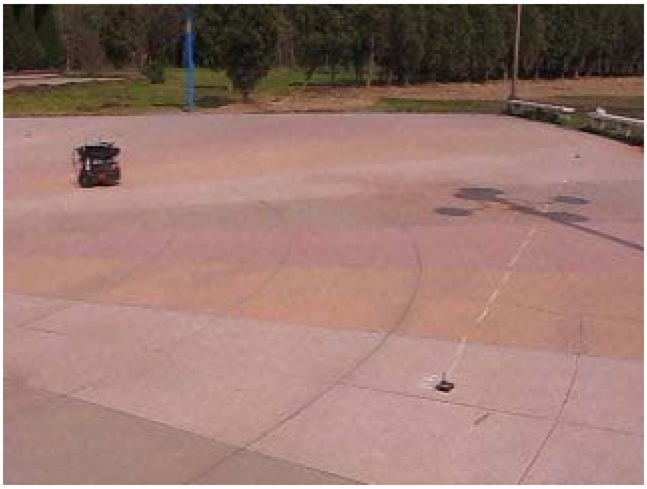


Fig. 22. Experiment field

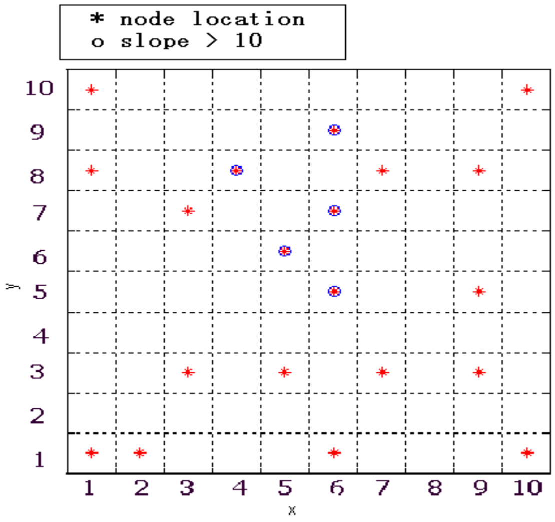


Fig. 23. Node distribution map.

20 sensor nodes are deployed in the field, four anchor nodes are deployed in four corners of the area. The coordinates of the anchors are (1,1), (1, 10), (10, 1) and (10, 10). Other nodes are placed randomly in a cell. All nodes are placed in the centre of the cells. The distribution of sensor nodes is shown in Fig. 23 .
In Fig. 23, + shows the location of a sensor node, while o means the slope of the sensor is bigger than 10◦. The experiment mission is to move the robot from the (1,1) cell to the (10, 10) cell, avoiding abstacles in the environment.

Real location(x,y)	Locating result(x,y)	Locating error (x,y)	slop(◦)
(1,1) *	(1,1)	(0,0)	2
(1,10) *	(1,10)	(0,0)	3
(10,1) *	(10,1)	(0,0)	3
(10,10) *	(10,10)	(0,0)	1
(6,9)	(5,9)	(-1,0)	75
(3,3)	(3,4)	(0,1)	2
(9,8)	(9,8)	(0,0)	0
(2,1)	(3,2)	(1,1)	4
(9,3)	(9,3)	(0,0)	0
(3,7)	(3,7)	(0,0)	1
(9,5)	(10,5)	(1,0)	2
(1,8)	(1,7)	(0,-1)	2
(6,5)	(6,5)	(0,0)	77
(4,8)	(4,8)	(0,0)	78
(5,3)	(5,3)	(0,0)	2
(7,8)	(8,8)	(1,0)	3
(6,7)	(6,7)	(0,0)	79
(7,3)	(7,3)	(0,0)	2
(6,1)	(6,1)	(0,0)	2
(5,6)	(5,6)	(0,0)	76

: * means anchor node.

Table 2. Node information

A. Using global path planning

Firstly, a series of experiments are carried out using global path planning to study the performance of the map building method mentioned before. The mobile robot communicates with the wireless sensor network and gets the environmental information. Using the map building method described above, we gets the environmental model as shown in Fig. 24. Set the slope threshold as 10◦, then gets the grid map in Fig. 25.

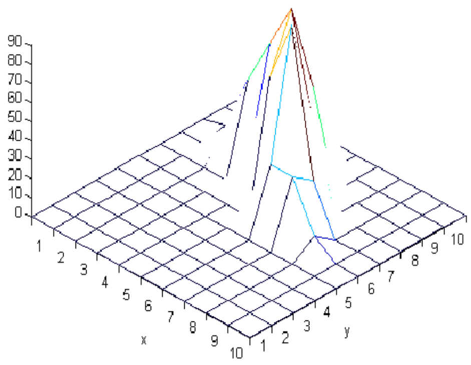


Fig. 24. Environmental model.

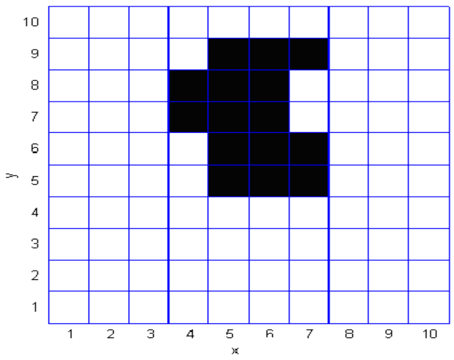


Fig. 25. Grid map.

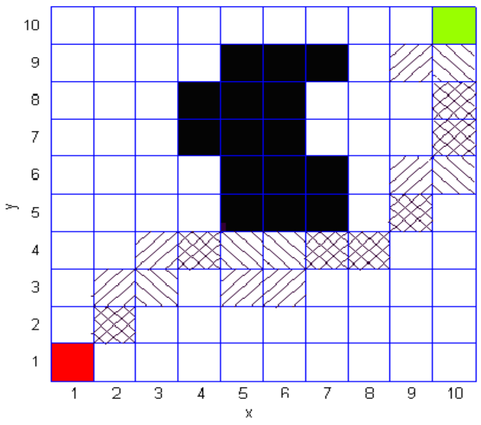


Fig. 26. Planned path and robot track in grid map.

The A* algorithm is used for path planning in the grid map. In order to move along the path, the robot gets its location with the help of WSN. Then the robot controls its move direction

according to the planned path and its location in real time. In Fig. 26, the cells with "\" mean the planned path. while the cells with "/" means the track of the robot. There are some cells with "x", which means that the robot track overlaps with the planned path.
The experimental results show that the map building method based on WSN performs well, and is applicable in mobile robot navigation system. From Fig. 26, we also see that there exists deviation between the planned path and the robot track, this is caused by the locating error.

B. On-line path planning

Now we adopt the on-line path planning method described above to study the performance of the system. Using the on-line path planning method, the robot doesn't need to acquire the global environmental information. Before navigation, the robot floods the information, including the destination and the "a" parameter in equation 8, to the whole WSN. Then, then sensor nodes calculate their cost to the destination according to their environmental information and distance to destination. During the navigation, the robot finds next landmark and move toward it. After reaching the landmark, the robot finds next landmark again, until it reach the destination. Fig. 27 shows the costs of different nodes in the 10x10 grid map where: a=40, upper threshold = 10°, lower threshold=1° , destination=(10,10).

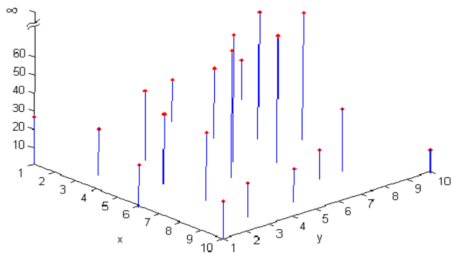


Fig. 27. Cost distribution (a=40)

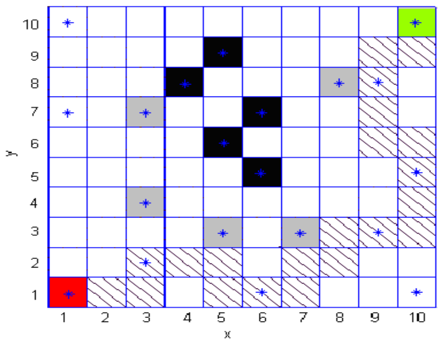


Fig. 28. Robot track in grid ma

In Fig. 28, "*" means sensor nodes, black cells are obstacles, cells with grey color have higher cost while white cells means free cell(with no sensor node). The cells with "\" shows the track of robot.
The experimental results show that the on-line path planning method proposed in section 3 succesfully achieve the navigation task, it's well suited for robot navigation applications. In the experiments, when using a smaller parameter "a", the robot trend to select a shorter path and move closer to obstacles. Sometimes it move into obstacle region which is caused by positioning error. By adjusting the parameter “a“, the Mobile Robot can make trade-off between safety and efficiency.

5. Conclusion

A mobile robot navigation approach based on wireless sensor networks is presented. WSNs can obtain various types of information such as the temperature, the humidity and the slope of the ground, and the proposed approach then used them to help the decision-making in navigation and path planning.
A WSN-based method for environment modelling and map building was proposed. This method utilized the distributed environment information obtained by the WSN to establish the environment model and the grid map with multiple attributes. Simulative analysis showed that the environment model established by this method achieved good match result on the map.
Subsequently, a dynamic monitoring function of WSN was utilized to adapt the Mobile Robot to the requirement of navigation on the changing environment. An on-line path planning method based on WSNs was proposed. Using this planning method, the Mobile Robot could make trade-off between safety and efficiency through adjusting the parameters used in the algorithm.
At last, an experimental WSN-based Mobile Robot navigation system was designed and implemented to verifying the proposed navigation approach. The experimental result showed that the proposed approach could fit the application requirement of Mobile Robot navigation.

6. References

J J Leonard, H F Durrant-Whyte. (1991) Mobile Robot Localization by Tracking Geometric Beacons [J]. IEEE Transactions on Robotics and Automation, 1991, 7(3):376-382
A. Davison, N. Kita. 3D simultaneous localization and map building using active vision for a robot moving on undulating terrain [C]. Proceedings of the IEEE International Conference on Computer Vision and Recognition. Hawaii, USA: IEEE, 2001:384-391
Chieh-Chih Wang,Thorpe.C Simultaneous localization and mapping with detection and tracking of moving objects [C]//Proceedings of the 2002 IEEE International Conference on Robotics and Automation. Washington D.C. USA: IEEE, 2002:2918～2924.
R Sim, N. Roy. (2005). Global a-optimal robot exploration in slam [C]//Proceedings of the 2005 IEEE International Conference on Robotics and Automation. Barcelona, Spain: IEEE, 2005:661～666.

Y. Bock and N. Leppard. (1990). Global Positioning System: An Overview, Spring Verlag, Berlin IAG Symposia Proceedings, Vol. 102, 1990.

Fenwick J.W., Newman P.M., Leonard J.J Cooperative concurrent mapping and localization [C]// Proceedings of the 2002 IEEE International Conference on Robotics and Automation. Washington D.C. USA: IEEE, 2002:1810 ~ 1817.

Burgard W., Moors M., Fox D., Simmons R., Thrun S. Collaborative multi-robot exploration [C]. // Proceedings of the 2000 IEEE International Conference on Robotics and Automation. San Francisco CA: IEEE, 2000: 476-481

Yamauchi B. Decentralized coordination for multi-robot exploration [J]. Robotics and Autonomous System, 1999: 111-118

Yahja A, Singh S, Stentz A. Recent results in path planning for mobile robots operating in vast outdoor environments [C].//Proceedings of the 1998 Symposium on Image, Speech, Signal Processing and Robotics. Hong Kong, 1998

Alan Mainwaring, Joseph Polastre, Robert Szewczyk, David Culler, John Anderson. Wireless Sensor Networks for Habitat Monitoring [C].// Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications. New York USA: ACM, 2002: 88 -97

Werner-Allen G, Johnson J, Ruiz M, Lees J, Welsh M. Monitoring Volcanic Eruptions with a Wireless Sensor Network[C]// Proceedings of the 2nd European Workshop Wireless Sensor Networks: IEEE, 2005: 108-120

Geoffrey Werner-Allen et al. Deploying a Wireless Sensor Network on an Active Volcano [J]. IEEE Internet Computing, 2006:18-25.

Guanghui He and Jennifer C. Hou. Tracking Targets with Quality in Wireless Sensor Networks[C]// Proceedings of the 13TH International Conference on Network Protocols, Boston, MA, United States: IEEE, 2005:63-7

Real-Time Wireless Location and Tracking System with Motion Pattern Detection

Pedro Abreu[a], Vasco Vinhas[a], Pedro Mendes[a],
Luís Paulo Reis[a] and Júlio Garganta[b]
[a] *Department of Informatics Engineering.*
Faculty of Engineering of Porto University
[b] *Department of Soccer.*
Faculty of Sport of Porto University
Portugal

1. Introduction

The Location and motion pattern detection concerning people or assets is an issue currently present in many business areas. Over the years, this task has been addressed under many different perspectives; however, due to their active consciousness, human beings are extremely unpredictable and so, the majority of these methods failed to provide accurate data that could be used for industrial purposes.

In the past years, tracking systems constituted an important research challenge for scientists and engineers in different areas of knowledge (e.g. computer science, sports, medicine, simulation, robotics), as well as industrial tracks. The ability of locating an object in a specific scenario in real-time and the capability to estimate its trajectory in a defined time frame constitute the two main goals and interests in this domain. Work in this field has gained a significant leverage in the recent past, mainly due to the emergence of consumer high performance computers, the development of low-budget high-quality tracking technologies, and the growing need for automated object location analysis. Over the past years, researchers in the robotic area still face two main tasks in this context: locating objects in the real world and relating that information with the corresponding item in a simulated environment, according to information retrieved throw robotic sensors; and using this data to apply navigation algorithms and methodologies that allow a robot to navigate on an unknown world. On the other hand, these tracking systems also find interesting applications on scenarios where the context environment is already known. In this situation, data mining techniques to detect motion patterns or behaviors assume a crucial role.

Nowadays, there are many types of tracking systems. All of them have their strengths and flaws regarding factors such as coverable area, occlusion problems, equipment costs, and so on. The most classical approach is camera-based (video surveillance systems). These solutions are usually cost-effective, with the advantage of providing complete digital storage and processing. Even though the video feed does not directly provide data on the positioning of entities, the use of image processing techniques will allow the extraction of entity positioning and additional information. Other tracking systems based on wireless emitter and receiver equipments emerged in the market with the advantage of providing clean data at a semantic level, displayable and storable in a straightforward manner, like GPS, Wi-Fi, RFID or Bluetooth, among others. The coverable area and the error involved are also important factors to be considered as well as the required resources, such as power and equipment density.

This research work proposes a system that takes advantage of typical redundant Wi-Fi networks and is based on a positioning engine built on top of these. This solution assumes that the client somehow carries an emitter device within the covered area. This system was tested using two distinct scenarios: a traditional retail indoor environment tracing clients through a commercial area; and in the sports area, more specifically in a soccer outdoor field, where this solution could constitute an important measurement tool for a professional soccer coach in the training sessions. The system presents a visualization platform of such data on real-time. The information can be displayed through several perspectives, including fully scalable concentration grids, clean positioning of the elements at hand or even a vision inference assuming that the items to be tracked are associated with people or robots. This application also works as a data collector, by storing appropriate information on a database. Using this datasets as a base, it is possible to reconstruct the paths taken by the elements and, therefore, predict and categorize typical routes and/or behaviors.

The remainder of this chapter is organized as follows: Section 2 describes the related work in the tracking system area. Section 3 presents all concepts involved in the system architecture that was proposed and their main functionalities. Section 4 exposes the results achieved and finally in the last section the conclusion are presented and future work trends are discussed.

2. Related Work

In the past years, tracking problems have constituted an important research challenge for scientists and engineers in different areas (e.g. computer science, sports, simulation or robotics, as well as industrial tracks). In the robotic area, for better modeling the world, it is extremely relevant to accurately process the signals received by the multiple sensors involved. Mapping real world objects to modeled ones is a critical task for the use of navigation algorithms and methodologies. Following these advances, the work published by Hyunwoong Park (Park et al.2006) presents a new kind of sensor system with multiple rotating range sensors. Such system allows a robot to guide itself on an unknown world. However, the ability of locating an object in a specific scene (where the context environment is already known) in real-time and the opportunity to estimate its trajectory in a defined time frame constitute the two main goals and interests in this domain. The emergence of consumer high-performance computers, together with the development of low-budget high-quality tracking technologies, and alongside with the growing need for automated object location analysis, provided the affirmative leveraged context for work in this field.

Nowadays, location and tracking systems represent a powerful tool in different scenarios such as:

- Consumer Pattern Recognition – detecting and representing a complete path of an object in a scene;

- Surveillance and Security – detecting suspicious behaviors or activities in a specific place;
- Autonomous Vehicle Navigation – helping in the prediction and avoidance of obstacles in a real world scenario.
- Sports Area – helping the club coaches optimize the performance of theirs athletes in a training session or during a match.

Despite the advances in this field, leading to noteworthy breakthroughs, some issues still remain to be addressed. From these, one ought to point out, in the first place, those concerning occlusion, which are classifiable in three distinct categories: self-occlusion, where part of the object to trace, typically articulated ones, occludes another part; inter-object occlusion where two traceable objects occlude each other; and occlusion by the background scene where the physical space's properties propitiate a camouflage of the object to track. For the inter-object occlusion research works like (MacCormick & Blake, 2000), (Elgammal et al., 2000) exploit the a priori knowledge of the object's position and attempt to predict possible occlusions and solve them smoothly.

Considering other domain issues, the hurdles that arise when tracking entities with non-linear movements must be addressed as this point constitutes one of the major problems in tracking persons in non controllable environments. A pragmatic approach to this situation could point out to time resolution diminishment in which tracking is achieved or the loosening of the real time requirements. Dealing with flexible layouts can also be problematic, as this situation may require constant system recalibration. This last step, in some cases, does not involve any level of reuse of the previously gathered data, making the whole process more time consuming than what it should be.

When the solution is based on video feed processing, there are some additional problems directly related to the inherent technology. The scene's illumination should be adequate so to facilitate the image binarization processes and the network, to transfer data from cameras to processors, should be enhanced and optimized in order not to become a system's bottleneck.

Summarizing in the next subsections a group of generic of-the-shelf and academic tracking systems shall be presented.

2.1 Generic Tracking Systems

In the literature there are many generic tracking systems that emerged over the past few years. These solutions are divided in two distinct groups: image based and non-image based

2.1.1 Non-Image Based Systems

Created by the US Government, the Global Positioning System (GPS) is a satellite-based solution, constituted in terms of hardware, by a twenty four satellite constellation with the ability to transmit all over the world. In the beginning, it was used solely by U.S military forces to aid in the planning of military operations. After the 80s this technology became available for the general public and so, many were the organizations that based their solutions on it.

Nowadays GPS is commonly used to perform real time detection of different types of vehicles and as a base tool to analyze their motion (Yu, 2005),(Nejikovsky et al., 2005). Yet in this scenario, the technology is applicable in three distinct ways: Cellular Based Tracking is a solution based in a conventional mobile phone with a GPS receiver that emits the vehicles position every five minutes. Wireless Passive Tracking has core advantage in using GPS, because once it is set up, there is no monthly fee associated, and with it is possible to collect information like for instance, how many pit stops are made by a vehicle in a given route and how fast is it moving. Although, Satellite Based Real Time Tracking is the most costly one, it enables different types of information collection such as long travel logs including details on duration, routes taken, pit stops, etc. Its worldwide coverable area constitutes an ideal solution for transporting companies.

The radio frequency identification (RFID) is a non-standardized wireless tag location method. This technology requires a RFID receiver and a set of tags which can be divided in two different groups: Passive - only detectable on a 13-meter radius from the receiver (e.g. new U.S.A. passports); Active Group - have their own internal power source, offer both reliable detection on a larger scale, and more resilience to occlusion problems caused for possible obstacles in the environment. The two major issues about this technology are the receiver's cost and the active tag's average unit price, mainly due to the need of an independent power supply (Chao et al., 2007).

Wi-Fi IEEE 802.11 technology allows establishing connectivity between a set of devices allowing an easy setup of wireless data networks in academic campus, industrial facilities, public buildings, etc. The technology behind these networks can also be used for designing a tracking system. By reusing commonly existing data networks and a low level protocol it is possible to create a tracking system on top of this infrastructure. Another advantage of this technology is the possibility of tracking an object by using a single access point, though the precision will weaken due to the lack of signal triangulation. Because of its technical details, the impact of issues such as occlusion and signals loss is reduced to a residual level especially in environments, which do not have high concentration of metallic materials (Mingkhwan, 2006).

Bluetooth is a wireless protocol available on any modern mobile equipment, allowing data exchange between multiple devices. It is exclusively used for short-range communications, which is the cause for its poor applicability on tracking systems. The battery consumption is also remarkably high (Jappinen & Porras, 2007). Despite the undeniable receivers' availability, the previously mentioned issues together with the non-transparent and possibly intrusive connection establishment process make this protocol inadequate for a efficient and reliable tracking system.

Some applications have infrared technology as a base for their location systems. The price attractiveness is one of the biggest advantages of it when compared with others, although there are no objective means to overcome occlusion issues especially in presence of opaque objects between the receiver and the target (Krotosky &Trivedi, 2007).

2.1.2 Image Based Systems

Thermal signature systems are one of the most expensive technologies for locating an object on a scene.

The main purpose of these solutions is the reconnaissance and processing of thermal images. These systems attempt to recognize specific thermal signatures of the entities being tracked although some items might not have them. Consequently the applicability of this approach is limited though the US Army made good use of it to train their night vision

dependent operatives (LaFollette & Horger, 1991) and biology researchers, to monitor fauna in the ocean (Raizer, 2003).
Multi-camera video surveillance is a technique that uses a set of cameras to track entities in a given environment. By accurately crossing the information coming from cameras which have intersecting frustums one can enhance the precision of the system, despite the possible processing overhead (Mittal & Davis, 2003). Camera calibration and its positioning on a tridimensional space may be performed/inputted manually (Collins et al., 2001), (Cai & Aggarwal, 1999) or even automatically despite the obvious errors that may occur if the last method is undertaken on an unsupervised way (Lee et al., 2000), (Khan & Shah, 2003). Although these systems are actually used in some scenarios, some issues still persist. The need to have a dedicated network for the system, the expenditure required for high-resolution equipment and the computational demands, are still major concerns.
In the literature, some research work tries to optimize the performance of these systems by minimizing the need of brute force computation (Mittal & Davis, 2003) and by using overlapping camera views (Huang & Russel, 1997), (Javed et al., 2003), (Kettnaker & Zabish, 1999).

2.2 Sports Video Analysis

One of the major research areas in the Colective Sport Games (CSG) is the sports video analysis. In football/soccer domain researchers had focus their work in problems like shot classification (Gong et al., 1995), scene reconstruction (Yow et al., 1995), structure analysis (Xie et al., 2004), (Xu et al., 2001), event extraction (Baillie & Jose, 2003), (Naphade et al., 1998) and rule-based semantic classification (Tovinkere & Qian, 2001). These approaches used the image transmitted by the television and recorded them for posterior processing (after the match ended).
These kinds of systems are categorized by Ekin (Ekin et al., 2003) in two main groups: cinematic and object-based ones. The object based uses algorithms to detect objects in a video while the cinematic uses features from video composition and produce rules.

2.2.1 Cinematic Approaches

Xu et. al (Xu et al., 2001) present a cinematic approach using for it the feature dominant color ratio to segment video. They defend that video reports should focus on play yard to extract game situations.
Xie et. al (Xie et al., 2004) used a Hidden Markov Models approach to detect two restricts events: play and break, in a video game. The complexity of this process is higher than in other sports like tennis or volley because for instance in soccer it is hard to determine if the game is stopped by a decision of the referee or by other highlights of the game-goal, corner, kick, shot, etc.
Other works like (Ren & Jose 2005) tried to expand Xie’s work and detect more game events like focus and replay in order to define new features/structures that they called Attack.

2.2.2 Object Base Approaches

The object base approach demands more computational resources but it allows more high-level domain analysis. In order to detect a large number of game events the work developed by Gong et. al (Gong et al., 1995) analyzes the ball’s trajectory and the relationship between the players' moves over the match. In the literature there are also many works that tried new approaches like merging audio and video information (Baillie & Jose, 2003). Although this kind of approach could constitute more high-level domain analysis one big issue is the asynchronies between audio and video queues.

2.2.3 Real Time Tracking Systems

Over the past few years new approaches appeared that use a multi camera tracking system to track players which promote new kinds of features like a near real time analysis. By comparing with classic approaches analyzed in the previous subsections, this system uses a fix number of stationary video cameras placed in a traceable environment. This type of approach increases the overall field of view reducing the dynamic occlusion problems and improves the accuracy and robustness of the information collected.

Cai and Aggarwal (Cai & Aggarwal, 1996) and Khan et. al (Khan et al., 2001) track the object using the best view camera and if the trackable object leads the field of view they change it to a neighbored camera. Other authors like Stein (Stein 1998) and Black (Black et al., 2002) assume that all trackable objects are in the same plane and compute the homography transformation between the coordinates of two overlapping images captured with uncalibrated cameras.

In the Xu et. al (Xu et al., 2004) work eight cameras were used and were calibrated in a ground plane coordinated system using Tsai's algorithm (Tsai 1986). Unfortunately this work presents some technical difficulties like problems with sparse landmarks in the coverable area that decrease the accurate calibration and data association and situations involving more than two players grouped in the same game region.

Summing, in CSG and more specifically in soccer the unique tracking systems that already exists in real environments are camera based. As demonstrated previously these systems still have to optimize some features like occlusion problems, computational demands, material cost and lack of portability.

2.3 Global Comparison

All the presented technologies have their optimal usage scenarios and with the purpose of choosing the best suited for this project a comparison using different parameters has been made.

For this matter six different parameters have been defined: cost involved which comprises the unit price per tag and the receiver's cost; accuracy that concerns the location error involved and the coverage area defined as the maximum area that an approach is capable of covering within acceptable values of the previous parameters. Energy consumption is also an evaluation parameter and it is especially relevant in technologies, where an external power source is required. Finally the response time is the time interval that goes from the acknowledgement of the last known good location of a tag and current one. Legal issues concern to the existence of legal aspects on system's implementation.

Also in this evaluation a scale with four distinct values (low, medium, high and very high) and two initials NA (not applicable) and A (applicable) are used.

By analyzing Table 1 one can conclude that Wi-Fi technology is the better option: it presents high levels of accuracy, in average less than 2 meters, it does not have any legal issues involved with its use and presents a very competitive hardware price. The possible reuse of an existing network infrastructure is another advantage to be taken into account since it has direct impact on the involved cost, despite the eventual need for a network strengthening to enhance triangulation possibilities. In addition to what was stated one could also mention that Wi-Fi is relatively immune to most of the occlusion that arise with other approaches.

	Costs		Accuracy	Coverage Area	Energy Consumption	Response Time	Legal Issues
	Tag	Receptor					
GPS	Medium	NA	Medium	Vey High	Medium	Low	A
RFID Passive Tag	Low	High	Low	Low	NA	Low	A
RFID Active Tag	Medium	High	Medium	Medium	Low	Low	A
Wi-Fi	Low	Low	High	High	Low	Low	NA
Bluetooth	Low	Low	Medium	Low	High	Low	A
Thermal Signature	NA	NA	Medium	Medium	NA	Low	NA
Infrared Sensors	NA	Low	Medium	Low	Low	Low	NA
Multiple Cameras	NA	Medium	Medium	High	Low	Medium	A

Table 1. Technology Comparison Table

RFID is the second best tracking solution analyzed. The use of active tags increases the coverable area – forty square meters using a single receiver – and increases accuracy levels. In spite of this obvious advantage, RFID still lacks standardization which naturally emphasizes the necessary integration process and supplier evaluation. Occlusion problems are not transparently solved and so there is some notorious interference in spaces with liquids and metals structures.
GPS's overwhelming worldwide coverage area conjugated with high levels of accuracy seamed to make this technology the elected one. Despite its advantages, its main purpose is for outdoor domains, instead of confined spaces where its relative accuracy is considerable lower. This fact associated with its significant tag cost relegates this technology as a third choice for indoor real-time tracking systems.
Although Bluetooth and infrared sensors represent good overall solutions in terms of cost, the coverage area is somewhat confined and the medium levels of accuracy invalidate the application of any of these systems in this project.

3. PROJECT DESCRIPTION

In this section, the undertaken project is described in detail in what regards its several components and analysis perspectives.
Having this in mind, and in order to have electricity support in the outdoor traced environment (the soccer field) an electrical infrastructure is detailed and after that the system's global architecture is depicted, in order to have an overall glimpse. In the indoor environment there is no need for an additional electrical infrastructure. The database model is further explained and the final three subsections are dedicated to the tool's individual description.

3.1 Electrical Infrastructure

Most of professional soccer coaches state that the training session should have the same length as a conventional soccer match-ninety minutes. Consequently any training support

system should stay active for all of this period. To fulfill this goal an electronic system was designed. In this approach a conventional 45A car battery is used directly connected to a 600w UPS. The UPS battery is also connected to the car battery in order to increase the autonomy of the system. This electrical infrastructure (Fig. 1) is capable to provide power for more than 120 minutes.
In order to increase the WI-FI network's density, a star topology approach is used. A router is connected directly to the battery's electrical extension and it is placed behind the goal. The access points (APs) were placed in specific points of the penalty box as shown in Fig. 1.

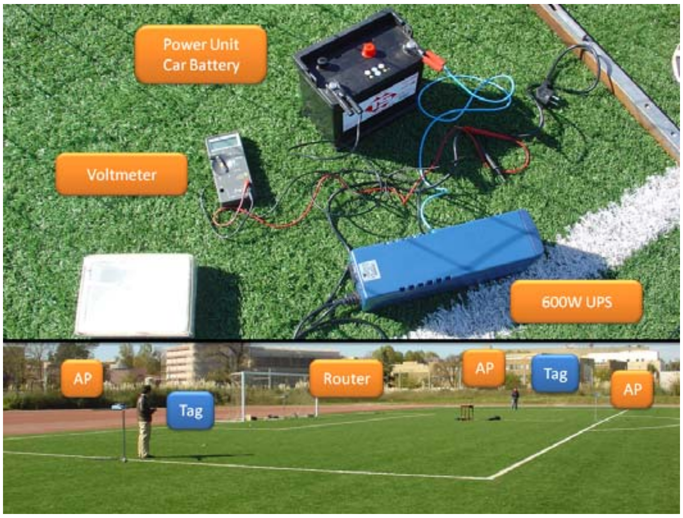


Fig.1. Electrical Infrastructure and WI-FI Network

3.1 Global Architecture

This subsection is dedicated to the illustration and depiction of the system's global architecture, concerning, not only its individual components, but, giving a particular emphasis on how they interact with each other and, therefore extract not only system module dependencies but also information flow analysis.
As revealed through Fig. 2, the elaborated technical design contemplates several independent modules that communicate in order to achieve a systematic unit.
Having the above mentioned in mind, and paying a closer attention to the numbers in figure, one is able to identify the system's modules as follows: Offline map editor; Wi-Fi enabled localization tag; Position Engine; Database for data integration and storage; Real-time monitoring application and Web enabled real-time and historical business intelligence.
Although most of these elements are object of further explanation in the next subsections, one ought to undertake a brief description of those whose nature is not obvious and, in order to, clarify their interaction.

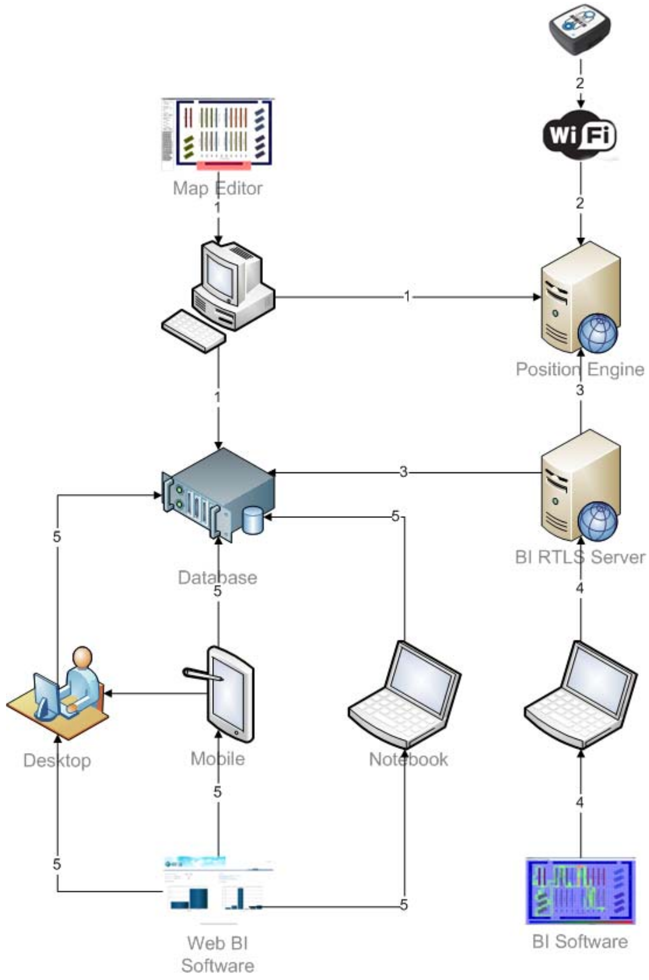


Fig. 2. System's Global Architecture

The first action, that ought to be conducted, in offline mode, consists in conducting a complete map creation\edition. The user shall specify, amongst other details, depicted in subsection Map Editor, the image file representing the shop floor or soccer field layout. For the first scenario, the user shall also specify the used scale and identify, by using a draw-like tool, what items are to be visible by visitors as well as spawn areas. This information is compiled in a XML file for both the position engine and real-time monitoring tool and submitted to the mentioned database for the historical BI application.
The Wi-Fi tag consists in a miniaturized active 802.11 a/b/g board with a couple of power batteries attached. These are configured to connect to a specific Wi-Fi network – security, DHCP but another network configurations are also possible – in order to directly communicate with the position engine. By using this kind of wireless technology, it is possible to reuse partially or totally the spot’s arena network infrastructure, having only, for

special requirement, a high density of access points as the accuracy naturally increases with this factor.

The used position engine periodically collects data from the tags and updates their position against a pre-loaded localization model. This model is very similar to the produced from the map editor differing only in the available information regarding visible objects. This model also requires a previous offline site survey for measuring Wi-Fi signal strength and for network items – routers and access points – precise localization. The engine is also web-enabled and supports a HTTP/XML API so that third-party applications can interact with it, therefore accessing localization and status information regarding each individual registered tag.

Using this communication protocol, the developed real-time monitoring server is responsible for gathering, at a specific periodicity – typically equals to the position engine frequency – every tag's valid location data. With this information, this module is directly responsible for updating the database and caching the session's data for the real-time monitoring application.

Having the continuous up-to-date database as a solid information reference, it was possible to enable both real-time and historical business intelligence applications. For real-time knowledge extraction, it was only used data referring to active sessions, for historical analysis, and delegating all the process effort to the database engine, specific and dynamic time windows were used to filter data. Despite the additional explanations that are given in subsection Real-Time and Historical BI Application, the versatility of such application must be referred as it congregates both web-enabled features and zero data process – it is all delegated to the database engine and allocated in a dedicated server – enabling its usage in a wide range of devices, including PDAs and mobile phones, alongside with traditional notebooks and desktop computers.

As a synopsis, one might refer the system's architecture as fairly distributed, where offline information regarding shop floor layout or soccer field layout, and wireless network definitions team up with a real-time web-enabled position engine, which enables third-party applications to collect and store data, so that diverse specific end-users can access both real-time and historical knowledge in a wide range of equipments, therefore enhancing management and coaching efficiency levels.

3.2 Database Model

In this subsection, the designed database model is revealed and justified.

Having into consideration the specific reported system's application in the retail and soccer domain – usually characterized with heavy data production and numerous simultaneous clients or for players movements all over the field respectively, combined with the project's idiosyncrasies – specifically in what concerns to localization tracking frequency – the database model paradigm followed consists in a hybrid form of a data warehouse star architecture with a slight normalization "flavor", as illustrated in Fig. 3.

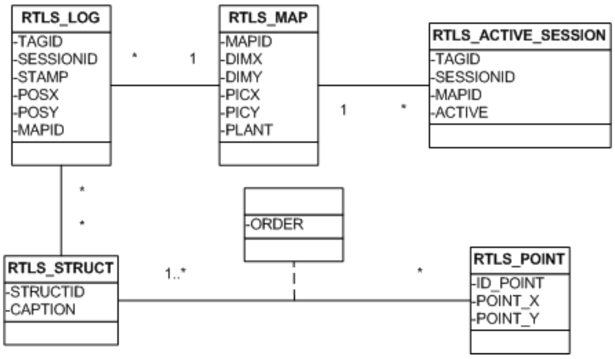


Fig. 3. Database Model

Regarding the strong star model, it is supported for the high data production levels, and perhaps most important, the fact that all data insertions are machine responsibility, as depicted in the above subsection, therefore preventing human error. It is also vindicated by historical analysis need that may cover hundreds of thousands and even millions of records.
On the other hand, some database normalization was introduced in order to cope with real-time requirements that would not be compatible with computing hundreds of records out of a table with millions of records, in a continuously systematic way. Another argument in favor of database normalization resides in shop floor or soccer field layout and visible structures definition.
Referring to specific database model items, one shall point the central relevance of rtls_log as central table responsible for storing all localization data. For each pair of tag/session identification, it is recorded each particular position in a given layout with a specific timestamp. The concept of session for the retail world is very similar to the concept of a client visit to a retail environment and, in an analogous way, the tag identification correspond directly to the physical entity, so that one particular tag can be used by several shop clients in non-overlapping time frames. On the other hand for the soccer scenario this concept may be different in each training session according to coach's decision. A new session could be related to three distinct situations: a player substitution (when a player is substituted by a colleague), a player out of the field (for instance to receiving medical assistance) or other situation when the player is out of the limits of the region that was defined by the coach for a specific situation in training session. In order to achieve real-time requirements, some redundancy has been introduced in what concerns active session identification, so that active shop floor clients or soccer players identification could be easily, and most important, efficiently retrieved. The main normalization features for the retail scenario, reside in layout definition – as it is stored one time in one table – as well as visible structures definition. These items can assume the shape of any kind of convex polygons and at each particular timestamp, a pair of tag/session can consistently perceive several items following the vision algorithm described in subsection Vision Algorithm.

3.2 Map Editor

In spite of the map editor tool being often referenced along this document, it is considered useful, for the sake of method replication, that it is further detailed, in this subsection.

As illustrated in Fig. 4, Map Editor is a traditional, network enabled, desktop application responsible for complete layout definition. The end-user (a shop manager or a soccer coach) shall open an image file and provide the interface with the drawing scale – in order to convert pixels to meters and vice-versa. Afterwards, the tool offers the possibility to pinpoint and draw, over the original layout, both visible structures (only in the retail scenario) and spawn areas – concept of regions where tag is hold between clients, e.g. cash registers, parking lots or entry/exit portals or concept that will allow the detection of new sections.

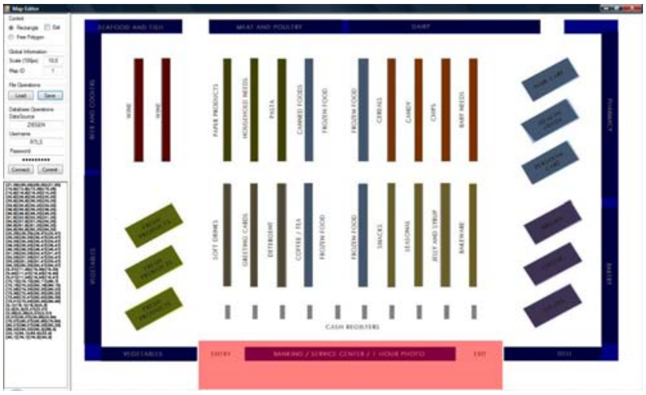


Fig. 4. Shop Floor Layout Complete Definition

Once the layout is completely defined, the user is able to save map characterization in a XML file in any available location and/or commit it to a specified database – with the previously described database model implemented.
The XML file will be an input for both the position engine and the real-time monitoring server, and, on the other hand, the committed database information is ground for historical computation and analysis.
Summarizing, the Map Editor constitutes itself as an auxiliary tool, vital for system's setup and dynamic enough to cover all the analyzed layouts. Its dual output enables a flexible usage for several system components and, simultaneously, due to XML openness, enables third-party development and integration.

3.4 Real-Time Monitoring Application

This system's module is, in conjunction with the web historical BI application, the project's core.
This unit is responsible for accessing location data from the online position engine and, simultaneously, using a multithread sliding-window approach, commit new data to the database and compute data into visual information following distinct approaches. Each of these tool's facets is mapped into a distinct GUI tab enabling independent analysis.
Before BI extraction, there are two mandatory configuration requirements that must be met: the first was already mentioned in previous subsections and consists in loading the layout; the second – except in debug mode – consists in establishing a HTTP connection with the position engine. The third, optional, requirement resides in opening a database connection

for online data insertion. If this is not met, there is a virtual infinite number of application's instances that can be run at the same time, enabling simultaneous BI extraction for numerous organization's members.
Taking into account, the enunciated tool's facets in what regards BI extraction, one shall pay closer attention to the aptitude of tracking, in real-time, all the clients present in the environment, alongside with their paths; demographic concentration with dynamic grid aperture; and visible structures concentration – these last two features have two modes of operation: strictly real-time and session history enabled.
Each of these tool's elements are subject of further description in the next subsections.

3.4.1 Real-time Tracking

This feature enables complete item tracking by overlapping current item's position directly over the loaded map information. This capability is independent of the GUI's windows size and/or shape as the coordinate systems are always synchronized.
As depicted is Fig. 5, it is also possible to enable session's path history directly over for instance a shop floor image, therefore enhancing visual perception of both current positions and session routes.

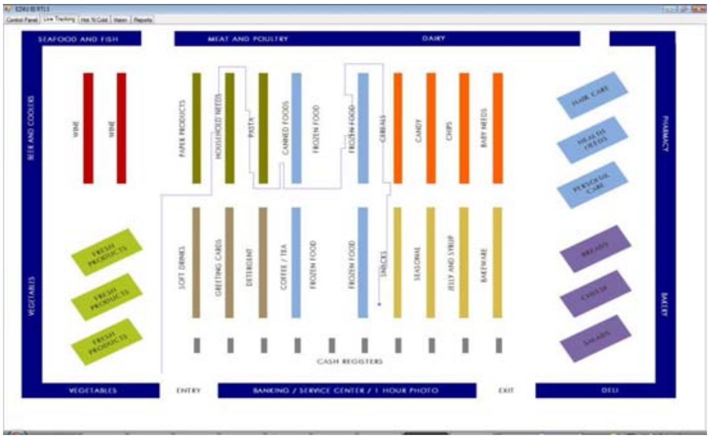


Fig. 5. Real-Time Tracking with Visible Path in a Shop Floor

In the presented screenshot, it is possible to see that at the time it was taken, there is only a single client/robot in the shop, whose location is near the layout center and that his visit concentrated mainly in direct routes in the north corridors.
If there would be more clients/robots present, it would be possible to perceive their current location by their blue dots representation as well as, optionally visualize their session routes.
This feature tries to emulate a bird's eye view of the all shop floor, with the possibility of recalling all the client's routes as if they left a visible trail while they are touring the facility.

3.4.2 Demographic Concentration

The demographic concentration feature enables space division using a completely dynamic, in terms of cell size, matrix grid that is colored according to demographic concentration at

given time. Once again, this computation can be performed using only strictly real-time positions or by recalling all current session's route positions.
Each cell is then colored following a three-color gradient where blue stands for lower levels of concentration, green for intermediate levels and red for high levels. A special remark is due to the fact that the entire color spectrum is used for the mentioned purposes.
As illustrated in Fig. 6, the concentration matrix is drawn, with a partially transparency, over the shop floor plant. In this practical example, the illustration corresponds to a situation where there were two clients/robots present in the shop, and it was selected, for concentration calculus concerns, not only current position but all routes' positions.

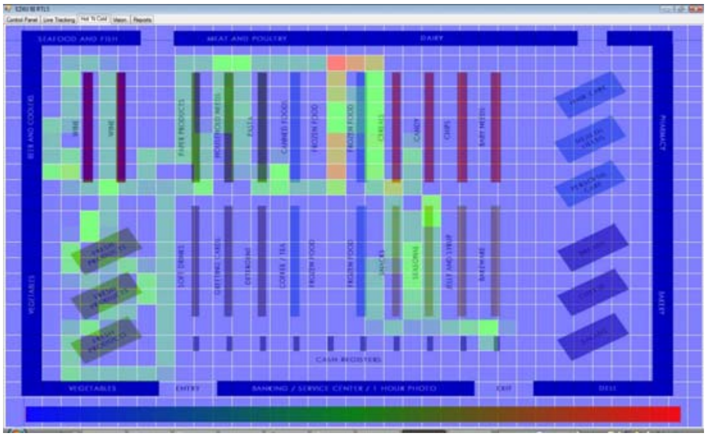


Fig. 6. Demographic Concentration with History in a Shop Floor

This tool's feature enables swift, yet effective, hot and cold zones analysis, current bottlenecks, unvisited versus most visited areas and online queue alerts and management. Also, in the soccer environment, this type of analysis specially the hot and cold zones ones, could represent good indicators for the principle coach measure their athletes performance in a specific training exercise. Also, as reported in the previous subsection, all the graphical information is independent of the GUI's window size or shape.

3.4.3 Vision Algorithm

In retail, in particular, and in many other domains, it is important, not only where the client is at a given moment, but also, and perhaps most significant, what is he seeing and/or paying special attention to (in the case of soccer, this analysis is focused on evaluating the decision-taking process by a player, which is related to his field vision in a specific time frame). Having this in mind, the authors decided to implement an efficient – due to real-time restrictions – yet effective vision algorithm, so that the defined visible structures would be able to contribute with a semantic business intelligence meaning.
Visible structures computation is based on client's orientation; conic field of view and occlusion calculus. Client's orientation is calculated having into consideration the last two valid locations. Using these two points, a simple vector is drawn from the previous valid position to the current point of observation. The conic field of view is projected in 2D, thus resulting in a parameterized triangle with flexible field of view and aperture angle. Once

shall also state that the world was slightly simplified and was considered to be a 2,5D reality, what in a retail practical example is admissible due to shelves' height being considerable greater than humans.
Having a valid point of observation and a triangular field of view defined, every potential visible structure identified in the given layout that intersects or is contained in the field of view is elected as a candidate for perception.
From this set of candidates, a proprietary occlusion detection algorithm based on trapezoid projections of the most apart vertices is run and those remaining candidates that are contained in this projected polygon are excluded from the set. After the algorithm conclusion, the remaining polygons are considered to be visible by the client.

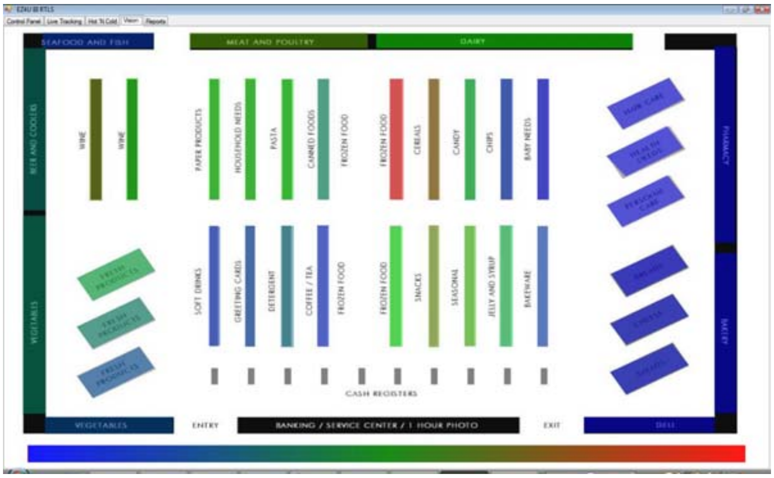


Fig. 7. Visible Structures Concentration with History in a Shop Floor

With this processing effectuated, a method similar to the one conducted in demographic concentration visualization, depicted in the last subsection, is performed. In other words, an absolute frequency auxiliary array is used to accumulate vision hits, each entry corresponding to a particular layout item. After the array update, to each item is assigned a color from the frequency spectrum – once again similar to subsection Demographic Concentration.
As Fig. 7 illustrates, the layout is now drawn in grayscale mode, and items are colored, in a semi-transparent way, representing the attention concentration. In the given example, it is clear that the top central shelves are the most viewed followed by the northwest corner while, on the other hand, the clients did not pay any attention to the selves on the right.
This vision concentration example is the outcome of the demographic concentration illustrated in Fig. 6.

3.5 Real-Time & Historical BI Application

In order to extract significant business intelligence knowledge based on historical data and not only real-time information, the authors decided to make an immediate use of the raw position data stored in the database.

Taking advantage of using Oracle as equally laboratory and production database, Oracle's Application Express was used to generate a web application responsible for processing data and, most important, aggregate information in an understandable way.

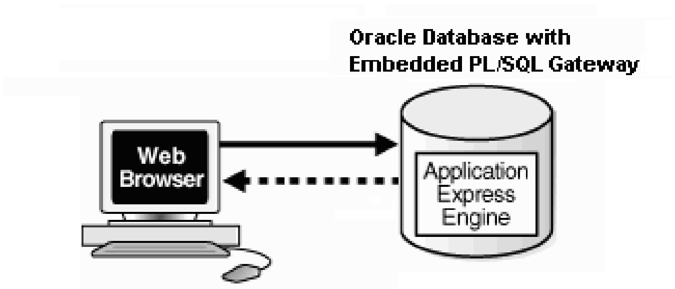
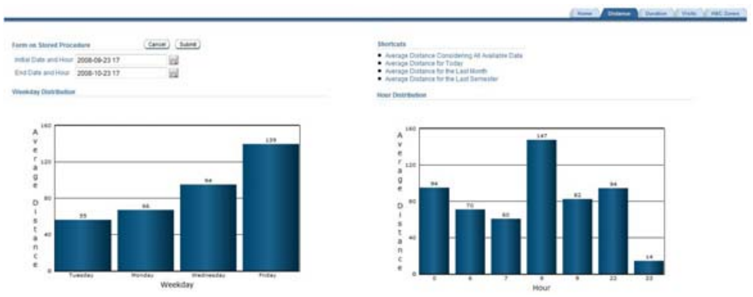

Oracle Database with Embedded PL/SQL Gateway
Web Browser
Application Express Engine

Fig. 8. Oracle Apex Embedded PL/SQL Gateway (Various Authors, 2008)

As depicted in Fig. 8, the Apex's engine is directly embedded to the database, thus directly dealing with client's web requests. With this architecture, several systems can easily access BI application as all heavy processing is database server's responsibility, leaving the client with only chart rendering computation.
Despite extensive tool's analysis being object of discussion in the next section, by using the described architecture and technology, several practical measures were considered for extraction, namely: hot and cold zones, average visit duration and distance and number of visits.


Form on Stored Procedure

Initial Date and Hour: 2008-09-23 17

End Date and Hour: 2008-10-23 17

Shortcuts

- Average Distance Considering All Available Data
- Average Distance for Today
- Average Distance for the Last Month
- Average Distance for the Last Semester

Fig. 9. Average Visit Duration Charts

All these benchmarks can be targeted to use only real-time data or recall specific historical time windows. A typical tool's screenshot is presented in Fig. 9, where a average visit duration charts are presented, considering data from the last month. On the left chart, data is aggregated by weekday and on the right by hour. Nevertheless, further details on this will be given in section 4.

4. Results

The obtained results are fairly positive both in terms of the developed infrastructure's operability and regarding the significance of the obtained data for analysts and managers.

On the first matter, one is able to state that the system performs swiftly as a whole as its constituting parts are able to exchange data harmoniously both in laboratory and production environments.

The first tool to be used in the entire process is the map editor. Its repeated use revealed that it is possible to fully sketch the plan of any layout with a coverable area that can go from just a few dozen square meters to hundreds of them, by the system being discussed in this document. The plan definition also includes allocating certain polygonal areas, as potentially visible structures, in the purpose of passing this information to the vision algorithm, mentioned in the previous sections. Doing so does not involve any remarkable additional complexity, in what concerns the tool's functionalities.

Surveying the space also proved to be an easy step to overcome, although more adequate equipment may be required for larger spaces, for instance when there is the need to use more than one calibration equipment simultaneously or when the space is large enough to justify the usage of an auxiliary trolley cart.

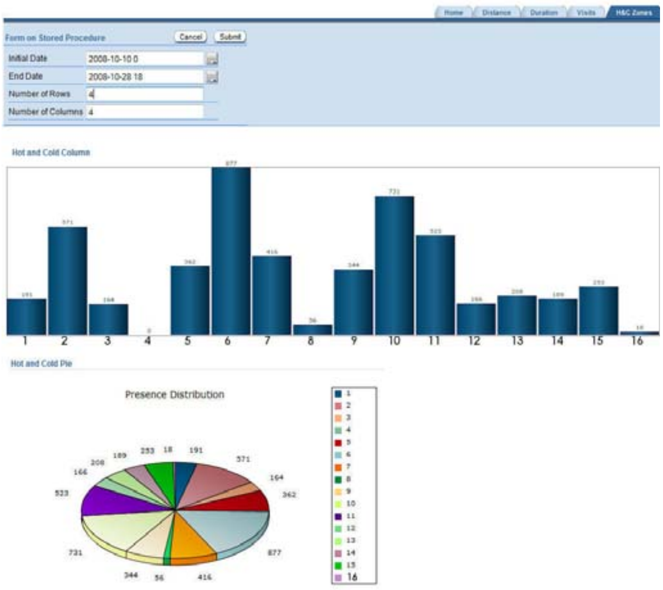
As for the core tool, the real-time monitoring application, it was observed that the displayed data is quickly interpretable, thus making its purpose: enabling real time business intelligence. In fact, just by looking, for instance, to the grid tab, managers are able to see what the current hot and cold zones are and make decisions based on such information. According to the retail store managers, whose identity was asked not to be revealed, by using the application one is able to, on a real time basis, assess the effectiveness of several decisions, namely: the store's layout, the number of open cash registers or even the work being carried out by employees in the zones where the customer service is made face-to-face.

The database model, used for the matter, proved to be adequate as the web application is able to present results within a maximum five seconds delay having the HTTP server and database service running on a machine with only two gigabytes of RAM. This can even be considered to be a poor server configuration for Oracle's 11g standards. The data in which these statements are based is an obfuscation of a real dataset and considers the central table of the star model to have about one million records.

Having already exposed the results obtained from the integration between the several parts of the system, in the remainder of this section some data collected from both scenarios will be presented and briefly discussed.

Fig. 10 represents the retail store grid concentration which has the layout as shown on Fig. 4. The data comprehends one week of activity where each chart entry corresponds to the number of accounted presences, on a specific zone, every two seconds. The zones are numbered from one to the number of rows times the number of columns. Zone 1 is located on the top left corner and the numbering grows a unit horizontally from left to right. All the cells are evenly distributed.

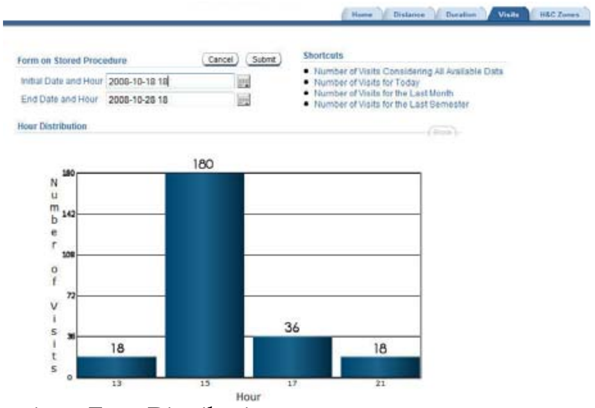
By analyzing the exposed charts, some interesting aspects on the customer's behavior when visiting the shop, can be assessed. For instance, although zones 13 and 14 correspond to the store's entry, it is observable that very little time is spent by the customers in those spaces. This aspect has economical relevance because all clients must past through the entry zone at least once, and there seems to be no products at that point to capture their attention. Another interesting point resulting from this analysis concerns the zones where more presences were accounted. All of them – zones 2, 6 and 10 – include central corridors, which are naturally part of the paths that clients have to cross to reach the products they are searching for. Zone 6's high value can also be explained by a promotion conducted by the shop on the products being sold there. On the opposite way, Zone 4 did not have a single client visiting it. This is probably explained by the poor attractiveness of the products present in that area as well as by the fact that its location is the farthest from the main entry point.



Form on Stored Procedure

Initial Date	2008-10-10 0
End Date	2008-10-28 18
Number of Rows	4
Number of Columns	4

Fig. 10. Web Application - Zone Distribution


Home
Distance
Duration
Visits
H&C Zones
Cancel
Submit
Shortcuts
Number of Visits Considering All Available Data
Number of Visits for Today
Number of Visits for the Last Month
Number of Visits for the Last Semester
Hour Distribution
180
18
36
18
0
36
72
108
142
180
Number of Visits
13
15
17
21
Hour

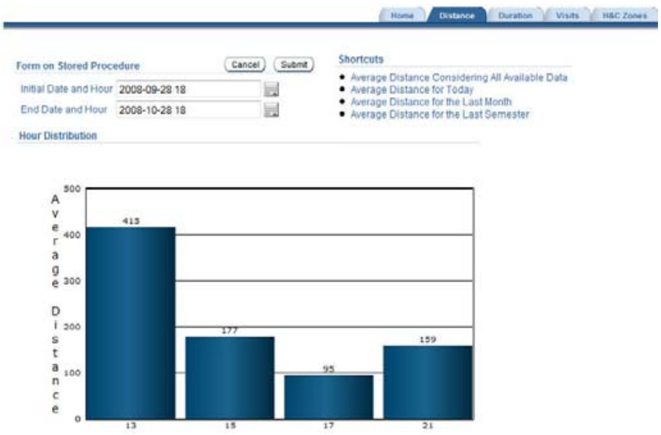
Form on Stored Procedure

Initial Date and Hour	2008-10-18 18
End Date and Hour	2008-10-28 18

Fig. 11. Web Application - Zone Distribution

The next evaluated measure concerns the number of visits made to the shop in the same time frame considered before. This data is represented on Fig. 11. Relating to this chart, one shall conclude that most of the clients that participated on this survey visited the shop at 3p.m.. The choice of conducting the survey at this time was optimal since this period is

already post lunch time and still before the rush hour that occurs usually at 5p.m.. Counter measuring this, the shop manager also tried to conduct the study at lunch and dinner time but the clients were not so cooperative in those periods and the overall affluence is also not so intensive compared to the first referred period. On full scale usage, the used tags shall be placed directly on the shopping carts, being completely transparent to customers.
Finally, the last type of information, discussed in the scope of this document, is the average distance walked by the clients, still considering the same time frame as before. By analyzing the chart in Fig. 12, it is evident that although there were few clients accepting to participate in the study at lunch time, those few walked about 400 meters within the store. Another interesting aspect is that the clients participating in study near the mentioned rush hour probably had very little time to shop since their walked distance is beneath 180 meters.



Form on Stored Procedure

Initial Date and Hour 2008-09-28 18

End Date and Hour 2008-10-28 18

Shortcuts

- Average Distance Considering All Available Data
- Average Distance for Today
- Average Distance for the Last Month
- Average Distance for the Last Semester

Hour Distribution

Fig. 12. Web Application – Number of Visits

Regarding the outdoor scenario tested in this project (the soccer field), a training exercise with four players involved was organized. The exercise's purpose was to train a player's shot accuracy after receiving a pass from a winger. In order to accomplished that, a goalkeeper, two wingers and a striker participated in this experience, having each of them a Wi-Fi tag attached to their shirts.
The penalty box was also divided in a 10*4 grid for calibration purposes and also to guide the site surveying process. The following picture (Fig. 13) exposes how the exercise was conducted.

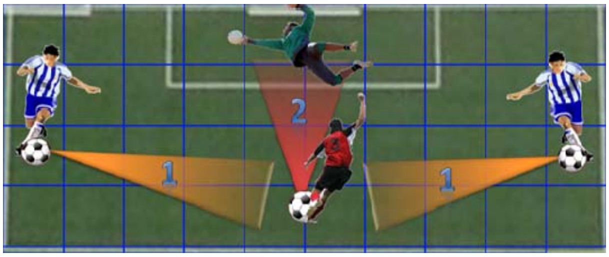


Fig. 13. Soccer Exercise Conducted

To clarify the Wi-Fi network's density one ought to first specify the access points' positioning. A router was placed behind the goal as well as the batteries and the entire electrical infrastructure described in the previous section. The remaining three access points were also used and positioned over the center of the remaining lines that define the penalty box (excluding the one which contains the goal line). To maximize the signal's strength all the Wi-Fi devices emitting a signal were put on top of a structure that allowed them to gain 1.20 meters of height. They were also put twenty centimeters away from the real lines so that the players' moves were not affected by their presence. Fig. 14 shows the signal's strength and noise levels on this particular scenario.

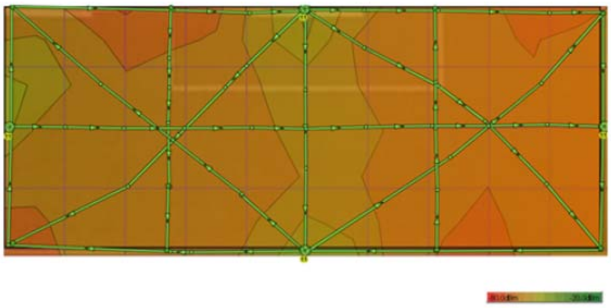


Fig. 14. Signal Strength and Noise for WI-FI Network

Since this is an outdoor environment the authors believe that the gathered noise values are the main cause for the error on the player detection because they are not being compensated by refraction and reflection phenomena which are typical in indoor environments. One ought to point out that this test was conducted with high-end devices and so there is a high probability of diminish the noise's impact just by changing the hardware to high-end artifacts, as their value mostly differs on the applied power on signal emission.

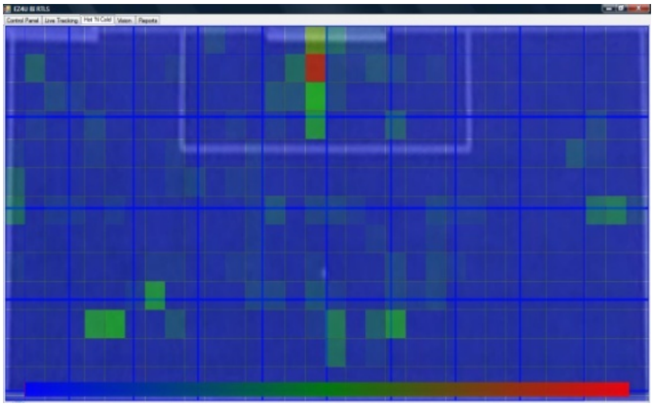


Fig. 15. Box density over an exercise

Even so, the next figures clearly demonstrate that the system was able to track the players during this exercise which lasted about thirty minutes. For instance, on Fig. 15, showing the

box's density over the entire exercise with the scale depicted at the bottom of the picture, one can observe a red cell on the goal area which undoubtedly corresponds to the goalkeepers' presence waiting for the striker's shots. The neighbor cells are also highlighted as the goal keeper moved a bit during the exercise in order to better cover the striker's shots on goal. The other highlighted cells demonstrate how the other three players moved during this training session.
Fig. 16 shows a real time screenshot of the player density where one can observe the wingers' position after having one of them pass the ball.

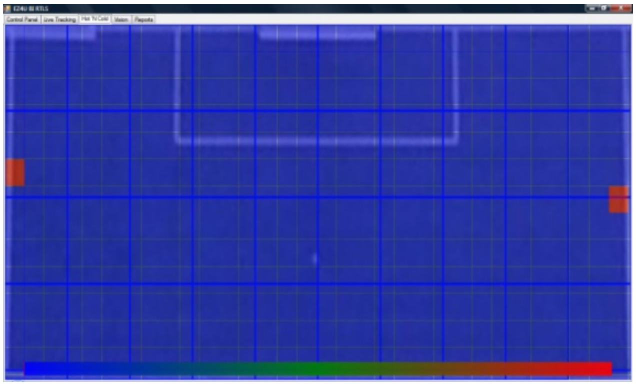


Fig. 16. Player density in the game field

And finally on Fig. 17, one can observe the left winger's and striker's position during a pass. On this particular figure the player's are represented as blue dots over the field. In this case the error between the obtained position and the real one did not exceed two meters for each player, which also justifies the fading green cells on the box's corner (shown on Fig. 15) as the wingers could decide from where they wanted to perform the pass as long as their distance to the box's limits did not overcome three meters.

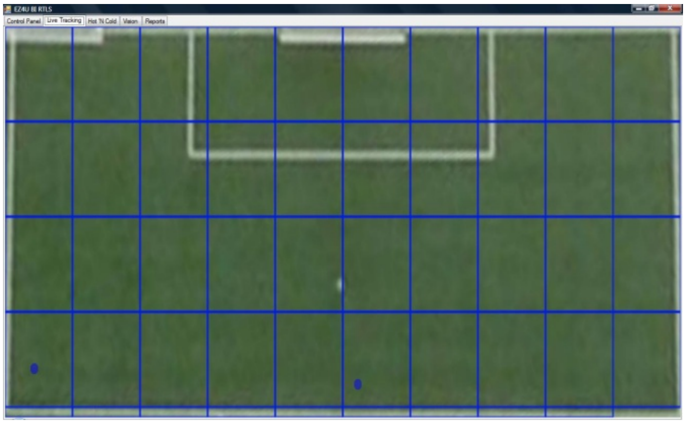


Fig. 17. Striker Position during a pass

Overall the system remained stable during the whole training session thus confirming its robustness and applicability as a tool for scientific soccer analysis.

5. Conclusions & Future Work

This section is dedicated to present and specify the project's main conclusions as well as identify and further detail major future work areas and potential collateral applications.

Admitting the first topic and having the conjunction between the project's module description, section 3, and its main results in the above section, one ought to affirm that all the most important goals were fully accomplished. In order to further support this statement, a brief hypothesis/result comparison shall be undertaken in the next few paragraphs.

First, a fully functional item real-time location and tracking system was pursued – without strict error-free requirements. The Wi-Fi based solution, not only complied to the specifications – real-time issues and non-critical error margin: less than 3 meters as maximum error – but did it reusing most of the client's network infrastructure (in the retail case) or using low brand equipments- router and access points (in soccer case). With this inexpensive tracking solution any team's coach has detailed reports about the performance of a specific player or the all team in a training session or even in a soccer mach. The possibility of having real time player positions in a specific situation and historical player paths constitutes an important tactical indicator for any soccer coach.

Secondly, the designed system's architecture proved to be reliable, efficient and, perhaps, most important, flexible enough to contemplate vast and diverse application scenarios. Also within this scope the distributed communication architecture performed as predicted enabling computation across distinct machines, therefore improving overall performance and reliability. This feature also enabled simultaneous multi-terminal access, both to the real-time analysis tool and the historical statistical software.

Taking into consideration the project's tools – real-time and historical – both were classified, by the retail company's end-users – mainly shop managers, marketing directors and board administrators and for sport experts - mainly clubs directors and academic experts – as extremely useful and allowed swift knowledge extraction, preventing them the excruciating, and not often useless – process of getting through massive indirect location data. The immediate visual information provided by the system proved to be effective in direct applications such as queue management and hot and cold zones identification, and most significant, in what concerns to visit's idiosyncrasy pattern extraction – as duration, distance and layout distribution – across different time dimensions, thus enhancing marketing and logistic decisions' impact. Also, in the sports area this system constitutes an important tool for measurement athletes' performance all over a training session.

Finally, in what concerns to direct results' analysis, one must refer to Oracle's APEX technology adoption. It has demonstrated to be able to allow multiple simultaneous accesses and, consequently, dramatically enhancing analysis empowerment, while, at the same time, eliminated heavy data computation from end-users terminals, concentrating it in controlled and expansible clusters. This characteristic allows through its web-based interface, accesses from unconventional systems such as PDAs, smartphones and not only notebooks and desktop computers. This particular feature is of great importance for on floor analysis and management and also for technical staff that for instance is spread through the soccer stadium in a match.

Regarding future work areas, and divided the two scenarios analyzed in this study, there has been identified a set of potential project enhancements that would be able to suppress some hurdles and, somehow, wide potential new applications.

For the retail environment, the first facet to be developed would be map edition oriented and should contemplate the possibility of defining multi-store and multi-location layouts in a single file. Also within this scope, it would be useful and technically straightforward – the definition of alarm/restricted zones where the entrance of a given tag or set of tags would trigger an immediate system response.

Secondly, considering business intelligence extraction, it would be useful to build or reuse an inference engine capable of determining the odds of a given customer turn right or left in the next decision point, taking for that, into account his past actions and comparing them to other customers' action that are classified in the same cluster. This aspect should be also applied to historical data so that efficient customer clusters would be defined and maintained.

Perhaps the most essential system enhancement would be the capability of, by dynamically change shop floor layout, and predict its impact in customers' routes and visits' parameters – duration, distance and financial outcome. This feature would make what-if scenarios possible to be run and immediate impact feedback would be given. Taking into account the current project's features and also the identified future enhancements, there have been identified several application domains that go beyond the retail segment.

In what concerns to soccer area one feature that could be interesting to explore as future work is the transformation of the actual system in a complete support decision framework for soccer coaches. For that purpose it is necessary to build a hybrid tracking system made by two synchronous modules. One module will be responsible for tracking the players and for this the actual system could be a solution and the other one should be responsible for tracking the ball. In this last problem one of two solutions could be adopted: a camera based classic solution with the advantage of only needing to track a specific object (with particular dimensions and color) decreasing so, the occlusion problems or adopt a new type of approach using, for instance, a chip inside the ball.

The second step for this new system will be the construction of soccer ontology. This point has particular importance because it helps to define concepts relationship with events of the game like: a pass, a shot, a corner etc. After that it is possible to construct a tracking system that will be capable to automatically detect game events, calculate historical player paths and in an advance face automatically detect player behavior relationship not only with their positioning but also with ball's. This system will definitely fill a gap in the market.

Taking into account the current project's features and also the identified future enhancements, there have been identified several application domains that go beyond the soccer or even CSG.

Amongst these, one shall mention the possible system's adoption by large warehouse management where traffic jams are not unusual. The proposed system would permit live vehicle tracking that in conjunction with a planning module would enable efficient traffic control, therefore avoiding bottlenecks, without compromising warehouse storage capacity.

Another possible application would reside in health care institutions where it would be useful for medical staff tracking around the facilities, in order to efficiently contact them in

case of emergency. Also within this domain, especially in mental institutions, patient tracking could be a great advantage.
Security applications are also easy to imagine, not only to track assets in a closed environment but also potential human targets such as children in public areas – such as malls hotels or conventional centers.
As a summary, it is fair to state that the project's initial ambitions were fully met and that the close cooperation with an important stakeholder in the global retail market and with an important university in the sports area was extremely important for better measuring the system's positive impact and potential firstly unseen applications. The technology transparency, allied with the future work areas, is believed to greatly improve potential applications in several domains, thus significantly widening the project's initial horizons.

Acknowledgements

The first and second author are supported by FCT (Fundação para a Ciência e a Tecnologia) under doctoral grants SFRH / BD / 44663 / 2008 and SFRH / BD / 36360/ 2007 respectively. This work was also supported by FCT Project PTDC/EIA/70695/2006 "ACORD: Adaptative Coordination of Robotic Teams" and LIACC at the University of Porto, Portugal.

6. References

Baillie, M. & Jose, J. (2003). *Audio-based Event Detection for Sports Video*, Lecture Notes in Computer Science, pp. 61-65. ISSN 1611-3349.

Black, J.; Ellis, T. & Rosin, P. (2002). Multi View Image Surveillance and Tracking, *Proceedings of IEEE Workshop on Motion and Video Computing*, pp.169-174, ISBN: 0-7695-1860-5.

Cai, Q. & Aggarwal, J. (1999). Tracking Human Motion in Structured Environments using a Distributed Camera System. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 21, No. 11, pp. 1241-1247, IEEE Computer Society.

Chao, C..; Yang, J. & Jen, W. (2007). Determining Technology Trends and Forecasts of RFID by a Historical Review and Bibliometric Analysis from 1991 to 2005. *Technovation - The International Journal of Technological Innovation, Entrepreneurship and Technology Management*, Vol. 27, No. 5, May-2007, pp. 268-279, Elvisier Ltd.

Collins, R.; Lipton, A. ; Fujiyoshi, H. & Kanade, T. (2001). Algorithms for Cooperative Multisensory Surveillance, *Proceedings of IEEE*, pp. 1456–1477, October.

Ekin, A. ; Tekalp, A. & Mehrotra, R. (2003). Automatic Soccer Video Analysis and Summarization. *IEEE Transactions On Image Processing*, Vol. 12, No. 7, pp. 796-807.

Elgammal, A.; Duraiswami, R.; Harwood, D. & Davis, L. (2002). Background and Foreground Modeling using Nonparametric Kernel Density Estimation for Visual Surveillance, *Proceedings of IEEE*, Vol. 90, No.7, pp. 1151–1163, ISSN: 0018-9219.

Gong, Y. ; Sin, L. ; Chuan, C. ; Zhang, H. & Sakauchi, M. (1995), Automatic Parsing of TV Soccer Programs.*IEEE International Conference on Multimedia Computing and Systems*, pp. 167-174.

Huang, T. & Russel, S. (1997). Object Identification in a Bayesian Context, *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, pp. 1276–1283, Morgan Kaufmann.

Jappinen, P. & Porras, J. (2007). Preference-Aware Ubiquitous Advertisement Screen, *Proceedings of the IADIS International Conference e-Commerce*, pp. 99–105, ISBN: 978-972-8924-49-2, Algarve, Portugal, December 2007, Sandeep Kirshnamurthy and Pedro Isaías, Carvoeiro.

Javed, O.; Rasheed, Z. ; Shafique, K. & Shah, M. (2003). Tracking Across Multiple Cameras with Disjoint Views, *Proceedings of Ninth IEEE International Conference on Computer Vision (ICCV)*, pp. 952–957, ISBN: 0-7695-1950-4, France, October 2003, Nice.

Kettnaker, V. & Zabih, R. (1999). Bayesian Multi-Camera Surveillance, *Conference on Computer Vision and Pattern Recognition(CVPR)*, pp. 117–123, IEEE Computer Society.

Khan, S. ; Javed, O. ; Rasheed, Z. & Shah, M. (2001). Human Tracking in Multiple Cameras. *International Conference on Computer Vision*, pp. 331--336.

Khan, S. & Shah, M. (2003). Consistent Labeling of Tracked Objects in Multiple Cameras with Overlapping Fields of View. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.25, pp. 1355–1360, IEEE Computer Society.

Krotosky, J. & Trivedi, M. (2007). A Comparison of Color and Infrared Stereo Approaches to Pedestrian Detection. *IEEE Intelligent Vehicles Symposium*, pp. 81-86, June 2007, Istanbul.

LaFollette, R. & Horger, J. (1999). Thermal Signature Training for Military Observers, *Proceedings of SPIE- Infrared Imaging Systems: Design, Analysis, Modeling and Testing II*, Vol. 1488, pp. 289-299.

Lee, L. ; Romano, R. & Stein, G. (2000). Monitoring Activities From Multiple Video Strams:Establishing a Common Coordinate Frame. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.22, No. 8, pp. 758–768, IEEE Computer Society.

MacCormick, J. & Blake, A. (2000). Probabilistic Exclusion and Partitioned Sampling for Multiple Object Tracking. *International Journal of Computer Vision*, Vol. 39, No. 1, pp. 57–71.

Mingkhwan, A.(2006). Wi-fi Tracker: An Organization Wi-fi Tracking System, *Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 213-234, ISBN: 1-4244-0038-4, May 2006.

Mittal, A. & Davis, L. (2003). M2 Tracker: A Multiview Approach to Segmenting and Tracking People in a Cluttered Scene. *International Journal of Computer Vision*, Vol. 51, No. 3, pp. 189–203.

Naphade, M. ; Kristjansson, T. ; Frey, B. & Huang, T. (1998). Probabilistic Multimedia Objects (MULTIJECTS): a Novel Approach to Video Indexing and Retrieval in Multimedia System, *Proceedings of IEEE Conference on Image Processing*, pp.536-540.

Nejikovsky, B.; Kesler, K. & Stevens, J. (2005). Real Time Monitoring of Vehicle/Track Interaction, *Proceedings of Rail Transportation Conference*, pp. 25–31.

Park, H. ;Lee, S. & Chung, W. (2006). Obstacle Detection and Feature Extraction using 2.5D Range Sensor System, *International Join Conference SICE-ICASE*, pp. 2000-2004.

Raizer, V. (2003). Validation of Two-Dimensional Microwave Signatures, *IEEE International Geoscience and Remote Sensing Symposium* (IGARSS), pp. 2694–2696, ISBN: 0-7803-7929-2, Vol. 4, July 2003.

Ren, R. & Jose, J. (2005). *Football Video Segmentation Based on Video Production Strategy*, Lecture Notes in Computer Science, Vol. 3408/2005, pp. 433-446.

Stein, G. (1998). Tracking from Multiple View Points: Self-Calibration of Space and Time. *Conference on Computer Vision and Pattern Recognition*. Vol. 1,pp. 1037-1042.

Tovinkere, V. & Qian, R. (2001). Detecting Semantic Events in Soccer Games: Towards a Complete Solution. *IEEE International Conference on Multimedia and Expo*.pp. 833-836.

Tsai, R. (1986). An efficient and accurate camera calibration technique for 3D machine vision, *Proceedings on CVPR*, pp. 323-344.

Various Authors. Oracle Application Express, Installation Guide Release 3.1.2 E10496-03, August 2008.

Xie, L. ; Xu, P. ; Chang, S. ; Divakaran, A. & Sun, H. (2004). Structure analysis of soccer video with domain knowledge and hidden Markov models, *Pattern Recognition Letters*, Vol. 25, No. 7, pp. 767-775.

Xu, P. ; Xie, L. & Chang, S. (2001). Algorithms and Systems for Segmentation and Structure Analysis in Soccer Video. *IEEE International Conference on Multimedia and Expo*. pp. 928-931.

Xu, M. ; Orwell, J. & Jones, G. (2004). Tracking Football Players with Multiple Cameras, *International Conference on Image Processing*. Vol. 5, pp. 2909-2912.

Yow, D. ; Yeo, B. ; Yeung, M. & Liu, B. (1995). Analysis and Presentation of Soccer Highlights from Digital Video. *Asian Conference on Computer Vision*. pp. 499-503.

Yu, Z. (2005). GPS Train Location and Error Analysis which Based on the Track Fitting of the Railway's Geometric Locus, *Proceedings of the Seventh International Conference on Electronic Measurement (ICEMI)*.

Sound Localization for Robot Navigation

Jie Huang
School of computer science and engineering The University of Aizu
j-huang@u-aizu.ac.jp

1. Introduction

For mobile robots, multiple modalities are important to recognize the environment. Visual sensor is the most popular sensor used today for mobile robots Medioni and Kang (2005). The visual sensor can be used to detect a target and identify its position Huang et al. (2006). However, since a robot generally looks at the external world from a camera, difficulties will occur when a object does not exist in the visual field of the camera or when the lighting is poor. Vision-based robots cannot detect a non-visual event that in many cases with sound emissions. In these situations, the most useful information is provided by auditory sensor. Audition is one of the most important senses used by humans and animals to recognize their environments Heffner and Heffner (1992). Sound localization ability is particularly important. Biological research has revealed that the evolution of the mammalian audible frequency range is related to the need to localize sound, and the evolution of the localization acuity appears to be related to the size of the field of best vision (the central field of vision with high acuity) Heffner and Heffner (1992). Sound localization enables a mammal to direct its field of best vision to a sound source. This ability is important for robots as well. Any robot designed to move around our living space and communicate with humans must also be equipped with an auditory system capable of sound localization.

For mobile robots, auditory systems also can complement and cooperate with vision systems. For example, sound localization can enable the robot to direct its camera to a sound source and integrate the information with vision Huang et al. (1997a); Aarabi and Zaky (2000); Okuno et al. (2001).

Although the spatial accuracy of audition is relatively low compared with that of vision, audition has several unique properties. Audition is omnidirectional. When a sound source emits energy, the sound fills the air, and a sound receiver (microphone) receives the sound energy from all directions. Audition mixes the signals into a one-dimensional time series, making it easier to detect any urgent or emergency signal. Some specialized cameras can also receive an image from all directions, but still have to scan the total area to locate a specific object Nishizawa et al. (1993). Audition requires no illumination, enabling a robot to work in darkness or poor lighting condition. Audition also is less effected by obstacles, so a robot can perceive the auditory information from a source behind obstacles. Even when a sound source is outside a room or behind a corner, the robot can first localize the sound source in the position of the door or corner then travel to that point and listen again, until it finally localize the sound source.

Many robotic auditory systems, similar to the human auditory system, are equipped with two microphones Bekey (2005); Hornstein et al. (2006). In the human auditory system, the sound localization cues consist of interaural time difference (ITD), interaural intensity difference (IID) and spectral change. Among them, ITD and IID cues are more precise than the spectral cue which, caused by the head and outer ears (pinnas), is ambiguous and largely individual dependent Blauert (1997). While the ATD and IID cues are mainly used for azimuth localization, the spectral cue is used for front-back judgment and elevation localization of spatial sound sources. The localization accuracy of elevation is far lower than azimuth.
In section 5, we describe a four-microphone robotic auditory system, three around the spherical head at the same horizontal plane with the head center and one at the top, for localization of spatial sound sources Huang et al. (1997b). The arrival time differences (ATD) to the four microphones were used to localize the sound sources. A model of the precedence effect was used to reduce the influence of echoes and reverberation. Azimuth or azimuth-elevation histograms were introduced by integrating the ATD histograms with the restrictions between different ATDs. From the histograms, the possibilities of sound sources can be obtained from the position of histogram peaks.
Comparing with other microphone array based methods Johnson and Dudgeon (1993); Valin et al. (2007), the array based methods use more microphones and need more computation power. While the array based methods are basically armed to obtain the best average of cross-correlation between different microphones, our method is based on the ATDs for different frequency components and different time frames. Since cross-correlation based methods calculate the cross-correlation function for all of the frequency components, when a sound source has high intensity than others, the low intensity sound sources will be easily masked. However, as we experience by our auditory system, we can usually localize sound sources with an intensity difference concurrently. It is because the different sound sources have different frequency components and appear in different time frames. By this mean, the histogram based method can have the advantage to distinguish sound sources with an intensity difference.

2. Bio-mimetic approach for Sound localization

When we design a sound localization system for a robot used in real environments, the tasks with first priority will be the robustness, high efficiency and less computation. It will be largely benefited by learning from the bio mechanisms.

High efficiency
Different animals will have different approaches to localize sound sources. For example, the humans use both ITD and IID cues for horizontal sound localization, and left the ambiguous spectral cue for elevation sound localization. It is because horizontal localization is the most important task the humans in daily
life.
The barn owls since need to localize both azimuth and elevation exactly in the darkness to hunt mice, they take a strategy to use ITD for azimuth localization and IID for elevation localization. Their right ear is directed slightly upward and the left ear directed downward Knudsen (1981). This up-down disparity provides the barn owls IID for elevation sound

localization. Insects like the bush crickets use their legs as acoustic tracheal system to extend the time and/or intensity differences for sound localization Shen (1993).
For a robot, it is possible to choose more than two microphones for sound localization. Since the ITD cue is the most precise one, if we use four microphones and arrange them at different directions covering the ITD cue for all of the directions, it will be highly efficient for the robot to localize sound sources in both azimuth and elevation.

Available for multiple sound sources
When there are multiple sound sources concurrently, we need some methods to distinguish them each other. The cross correlation method is a most popular method for time difference calculation. It gives the similarities between two signals for different time disparities, and we can find the most similar point by its peak position. By this method, we usually need to analyze signals for a relitave long time period to identify multiple sound sources. Moreover, since the cross correlation method is a averaging method for all of the frequency components, if there are two signals with different intensities, the smaller signal will be masked by the louder signal.
In the human auditory system, the time difference is calculated by local information, limited in both time and frequency range. The local time difference information is then integrated by an histogram-like method, with the weights not only depends on its intensity but also effected by the precedence effect, to find out the correct ITDs (characteristic delays) for different sound sources. This method can distinguish multiple sound sources even with intensity disparities.

Robustness
Since many robots are to be used in the human daily environments, the robustness for echoes and reverberation is very important for robotic auditory systems. As in the human auditory system, the robotic auditory systems also need to incorporate the precedence effect. By the EA model of the precedence effect, we can calculate the weights depend on the estimated sound-to-echo ratios to give more priority to echo free onsets to reduce the influence of echoes. The precedence effect not only reduce the influence of echoes, but also can increase the separation rate for multiple sound sources. It is because the echo free onsets are usually the parts of the beginning of a sound, or the parts where the sound level increases sharply. In all cases, the sound intensity is contributed by a single dominant sound source.

3. Sound localization cues in the human auditory system

3.1 Cues for azimuth localization
Sound localization is to identify the direction, including azimuth and elevation, of sound sources by the information obtained from the sound signals. For horizontal sound sources, it is well known that the interaural time differences (ITD) and interaural intensity differences (IID) are the most important localization cues Blauert (1997).
The ITD cue is caused by the arrival distance disparity from sound source to the two ears. When the distance of sound source is far away from the head $(D \gg d)$, the incidence of sound is parallel and the arrival time difference can be approximated as

$$\Delta t = \frac{d}{2}(\theta + sin\theta) \; -\tfrac{\pi}{2} \leq \theta \leq \tfrac{\pi}{2} \tag{1}$$

where d is the diameter of head and θ is the azimuth of sound source.

In the human auditory system, the time disparities of a sound signal is coded by the neural phase-lock mechanism, i.e., auditory neurons fire at a particular phase angle of a tonal stimulus up to about 5 kHz Gelfand (1998). For signals have frequency components of more than about 1.5kHz, where the wavelength becomes shorter than the distance between the two ears, the time difference information can not be recovered from the phase difference uniquely because of the phase wrapping phenomenon.

Suppose the phase difference of frequency component f is $\Delta\psi_f(-\pi < \Delta\psi_f \leq \pi)$, the possible real phase difference $\Delta\hat{\psi}_f$ can be

$$\Delta\hat{\psi}_f(n) = \Delta\psi_f + 2\pi n_f \tag{2}$$

where $n_f$is an integer depends on each frequency.

Biological studies about owl's auditory system revealed that the phase difference is detected by a neural coincident detector Konishi (1986), and the reduction of redundancy is done by finding out the characteristic delay (CD) Takahashi and Konishi (1986), the common time difference among multiple different frequency components.

For sound signals containing multiple frequency components, the task is to find an integer n_f for each frequency component f, so that the time difference $t_{CD}$is the same for all frequency components

$$\Delta t_{CD} = \frac{1}{2\pi f}\Delta\hat{\psi}_f = \frac{1}{2\pi f}(\Delta\psi_f + 2\pi n_f) \tag{3}$$

On the other hand, the IID cue, caused by the shadow effect of the head and outer ears, is significant for high frequency sounds but becomes weak as the frequency decreases. It is large when the sound comes from side directions (left and right) and small when the sound is from front and back. It is more complex to formulate the intensity difference compared to the time difference. Addition to the interaural cues, the spectral cue is used to disambiguate frontback confusion of the ITD and IID cues Blauert (1997).

3.2 Cues for azimuth and elevation localization

For sound sources in the 3D space, interaural cues are not enough for both azimuth and elevation localization. For example, the possible source positions which create the same ITD to the two ears will be approximately a locus of conical shell which known as the cone-of-confusion in psychoacoustic studies Blauert (1997). The changes of spectral characteristics are important for elevation localization. For example, sound sources in the median plane will not create any interaural difference (assume the auditory system is left-right symmetry), sound localization is mainly due to the spectral cues.

The directional spectral changes can be represented by the transfer function from the sound source to the two ears, the so-called head-related transfer functions (HRTFs). The frequency characteristics of HRTFs, influenced by the head, ears, shoulder and body, are variant with azimuth θ and elevation ρ. By the spectral changes together with ITD and IID cues, the auditory system can identify the azimuth and elevation of a sound source concurrently. However, compared to the interaural cues, the spectral cues are weaker, individual dependent, and easy to be confused.

In general, the HRTFs contain not only spectral cues, but also interau-ral cues. Representing the HRTFs for left and right ears as $Hi(9,ip,f)$ and $H_r\ (9,ip,f)$, the cross interaural transfer function can be defined as

$$H_x(\theta,\rho,f) = \frac{|H_l(\theta,\rho,f)|}{|H_r(\theta,\rho,f)|} e^{jp(\theta,\rho,f)} \tag{4}$$

or the opposite $(H_x = H_r/H_l)$, where p is the phase difference. The amplitude part of H_x provides the ITD, and the group delay of the phase part provides the IID information.

4. The Echo-Avoidance Model of the Precedence Effect

4.1 Introduction

When a sound is presented in a reverberant environment, listeners usually can localize the sound at the correct source position, being unaware and little influenced by the surrounding reflections. This phenomenon, including its different aspects, is referred to by different names, Haas effect Haas (1951), Franssen effect Franssen (1959), the first front effect Blauert (1997) and the precedence effect Wallach et al. (1949).

The precedence effect has been a topic of continuous theoretical interest in the field of psychoacoustics for more than half a century Gardner (1968). Evident from developmental psychological studies suggest it is a learned effect in the human auditory system to cope with echoes in ordinary reverberant environments Clifton et al. (1984). Because humans spend much time indoors in a typical reverberant environment, the needs for a human to localize sound may cause the human auditory system to adapt to reverberant environments.

Recent studies also show that the precedence effect is active and dynamic. Blauert and Col Blauert and Col (1989, 1992), Clifton and Freyman Clifton (1987); Clifton and Freyman (1989) and Duda Duda (1996) reported that the precedence effect can break down and become re-established in an anechoic chamber or hearing test by headphones.

The precedence effect is an important factor for acoustical architecture design Haas (1951) and stereo sound reproduction Snow (1953); Parkin and Humphreys (1958). It is also important for computational sound localization in reverberant environments Huang et al. (1995, 1997b). Moreover, since the precedence effect influences the spatial cues in reverberant environments, it is also important for the perceptual segregation and integration of sound mixtures, the so-called cocktail-party effect Blauert (1997); Bodden (1993); Cherry (1953), or auditory scene analysis Bregman (1990) and its computational modeling Cooke (1993); Ellis (1994); Huang et al. (1997a); Lehn (1997).

Despite the large number of psychological studies on the precedence effect, there have been few computational modeling studies. Some abstract models, e.g. funneling models von Bekesy (1960); Thurlow et al. (1965), inhibition models Haas (1951); Harris et al. (1963); Zurek (1980, 1987); Martin (1997) and others McFadden (1973); Lindemann (1986a,b); Litovsky and Macmillan (1994) have been proposed for the precedence effect. Basically, while the funneling models proposed that the localization of succeeding sounds is biased toward the direction which has been established by the first-arriving sound, the inhibition

models argued that the onset of a sound may trigger a delayed reaction which inhibits the contribution of succeeding sounds to localization.
In all those models, the precedence effect is considered to be triggered by an "onset". Zurek argued that the onset should be a very "rapid" one, but no quantitative criterion was given for a rapid onset. Furthermore, neither funnel-ing nor inhibition models provide a consistent explanation for psychoacoustic experiments with different types of sound sources.
According to the Zurek model, the inhibition signal takes effect after a delay of about 800μs, and lasts for a few milliseconds. The inhibition interval was determined based on the just-noticeable difference (JND) tests of interaural delay and intensity judgment which showed that the JND level increases in the interval range from about 800μs to 5 ms. However, the psychological experiments conducted by Franssen indicated that the sound image of constant level pure tone was localized by the transient onset and could be maintained for a time interval of seconds or longer Franssen (1959); Hartmann and Rakerd (1989); Blauert (1997). Other psychological experiments, e.g. those by Haas (1951) using speech and filtered continuous noise, have shown that the inhibition occurs after a time delay of about 1 ms to about 50 ms according to the type of sound source used in the tests.
The Zurek model cannot distinguish the different phenomena caused by different types of stimuli. One more point to be noted is that the inhibition in the Zurek model was absolute, i.e., a very small onset can inhibit any high-intensity succeeding sound. This obviously conflicts with the fact that the precedence effect can be canceled by a higher-intensity succeeding sound.
A computational model on the precedence effect must give a systematic interpretation of the results of psychological tests and provide a theoretical explanation for the phenomenon.
Because of the needs for human to localize sounds in reverberant environments, it is our opinion that there should be a mechanism which can estimate the level of reflected sounds and emit an inhibition signal to the sound localization mechanism, so that the neural pathway from low to high level of localization processing can be controlled to avoid the influence of reflections. Such a mechanism is possibly located in the cochlear nucleus Oertel and Wickesberg (1996). From this point of view, the precedence effect can be interpreted as an "echo-avoidance" effect. Here as well as later, the term "echo" is used with the wide meaning of all reflected sounds by the surrounding.
In this section we will propose a new computational model of the precedence effect, the Echo-Avoidance (EA) model (Section 1.4.2), with an echo estimation mechanism. We will show that the EA model of the precedence effect can be used to detect available onsets which are relatively less influenced by echoes. The model can explain why the precedence effect occurs in transient onsets and can interpret the data obtained by several psychological experiments consistently
(Section 1.4.3).

4.2 The Computational Echo-Avoidance (EA) Model

The EA model of the precedence effect, similar to the Zurek model, consists of two paths, one for localization cue processing and one for inhibition signal generation as shown in Fig. 1.1. We assume that the echo estimation and inhibition mechanism is independent for different frequencies Hafter et al. (1988). Both binaural and monaural localization processes are effected by the precedence effect Rakerd and Hartmann (1992). In the integration process, averaging for different localization cues in all of frequency subbands will take place.

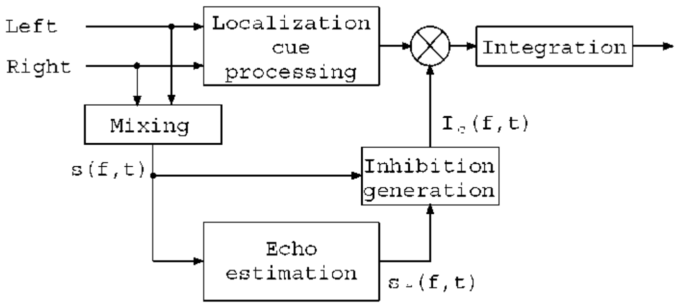


Fig. 1. EA model of the precedence effect.

When an impulsive sound is presented in a reverberant environment, the resulting signals arrive our ears are first the direct sound and then followed by a series of reflections. The sequence of reflections depends on the shape and reflection rates of the surfaces in the environment, and the position of sound source and observation points.
It is impossible for the auditory system to distinguish all of the reflections one-by-one. The sound image we perceive is a series of sound impulses whose amplitudes decay over time in an minus exponential manner approximately. Thus, the auditory system may learn about two features, decay and delay, related to the reflections.
These two features can provide a prospective pattern of echoes:

$$h_e(t) = \{\begin{cases} 0 & ,0\tau_0 \\ ke^{-(t-\tau_0)/\tau_d} & ,t \geq \tau_0 \end{cases} \tag{5}$$

where k, τ_0 and τ_d are learned parameters, correspond to the strength and delay time of the first reflections, and the time constant of decay respectively.
Denote the sound level of a particular frequency by $s(f,t)$. Thus, the possible echo can be estimated as:

$$s_e(f,t) = \underset{t}{Max}\{s(f,t-t')h_e(t')\}, \qquad \tau_0 < t' < \infty \tag{6}$$

Here, the operator Max, instead of sum, is to take the maximum value for all of t', since $h_e(t)$ is not a real impulse response but an approximation pattern.
An abstract illustration of the relation between received sound and estimated echoes is given in Fig. 2(a). The signal is represented in discrete time with an interval t_s which can be the sampling interval or the length of a time frame. By the exponential decay feature in the echo estimation mechanism, the algorithm

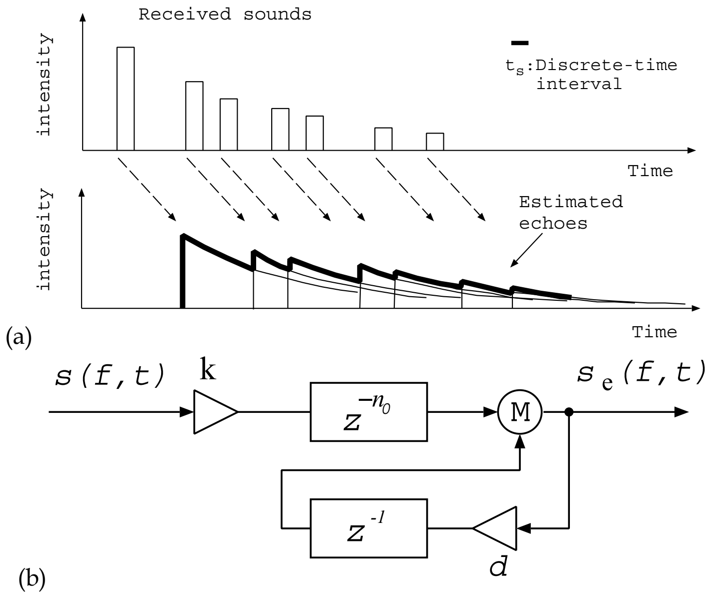


Fig. 2. (a) An abstract illustration of the relation between received sound and estimated echoes. (b) A feedback algorithm for echo estimation.

can be implemented using a feed-back algorithm as shown in Fig. 2(b), where $n_0 = \tau_0/t_s$ and $d = e^{-t_s/\tau_d}$.

An example of echo estimation and echo-free onset (see Section 1.4.3) detection is shown in Fig. 1.3. In this figure, the last plot shows the positions of echo-free onsets given as $s(f,t)/s_e(f,t) \geq 3$ (The value 'three' of sound-to-echo ratio is not critical. From our experience, a weight of 0.1 to 1 corresponding to sound-to-echo ratios 2.1 to 3.0 were used for averaging in sound localization algorithm.) It is clear that each echo-free onset corresponds to a sharp increase point in amplitude, and the sound level before the echo-free onset is relatively low. Another clear fact is that the echo-free onset does not depend on the absolute strength of the sound.

Fig. 1.4 shows the influence of echoes on arrival time differences for a time period around echo-free onsets detected by the EA model. In this figure, the horizontal axis is time with the origin about 0.03 second before each detected onset. The first plot is the average amplitude of sound signals arround the echofree onsets. The second plot is the average magnitude of estimated sound-to-echo ratios. The third plot shows the distribution of time differences between a microphone pair for every echo-free onset. The last (fourth) plot is the standard divergence of the time differences (the dashed-line is the real time difference for reference.) From the results, we can see that the standard divergence of time differences is remarkably smaller near the onsets than in other portions. This demonstrated that the onsets detected by the EA model are indeed less.

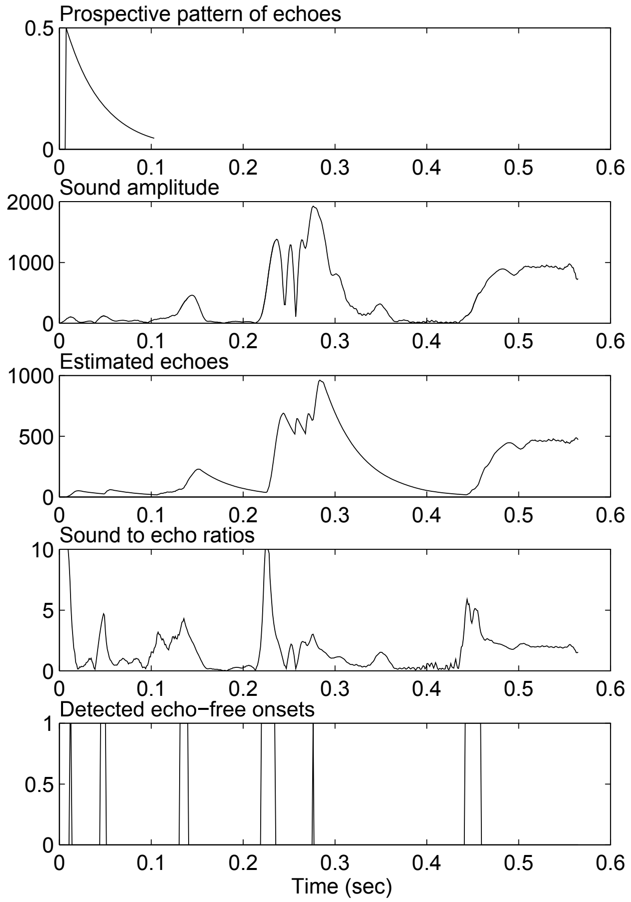


Fig. 3. An echo-free onset detection test by the model based algorithm. The first plot is the prospective pattern of echoes $h_e(t)$. The second plot shows the amplitude envelope of a test sound $s(f;\ t)$ (filtered speech signal, central frequency 255 Hz and bandwidth 30 Hz). The third and fourth plots show the estimated echoes $s_e(f;\ t)$ and the sound-to-echo ratios $s(f;\ t)=s_e(f;\ t)$. The last plot shows the positions of echo-free onsets as $s(f;\ t)=s_e(f;\ t) \geq 3$.

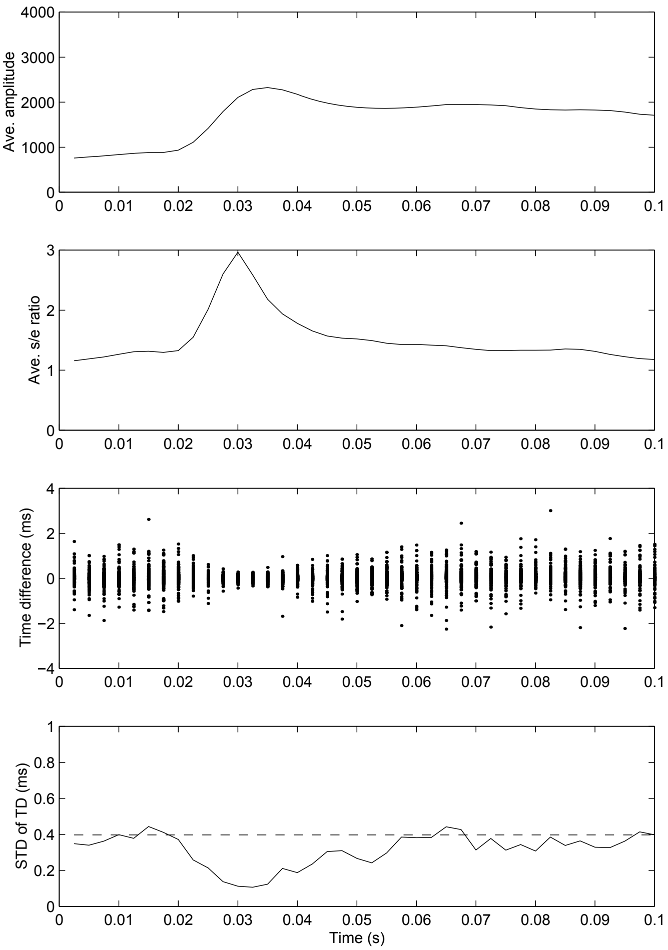


Fig. 4. The time variance of phase difference around onsets (speech source, ordi-nary room, 5 s length, 137 detected onsets).

4.3 Psychological Experiments and Their Explanations Increase of just-noticeable difference

Zurek Zurek (1980) reported that the just-noticeable difference (JND) of interaural delay and intensity difference, tested by headphones, increased temporarily following a preceding burst sound. The time period of the increase ranges from 800 / s to 5 ms after the preceding burst. Saberi and Perrott Saberi and Perrott (1990) reported similar results by a different method. They showed the perceptual thresholds of interaural time difference increased after the preceding click sound in a time range from 500 / s to 5 ms. The results of Saberi and Perrott' experiments are shown in Fig. 1.5 (reproduced from FIG.3 in Saberi and Perrott (1990)). In this figure, the 'x' marks show the increase of the perceptual threshold of interaural time difference. We can see that the increase pattern of the perceptual threshold of the interaural time difference also has similar features (delay and decay) as we mentioned in the prospective pattern of echoes.

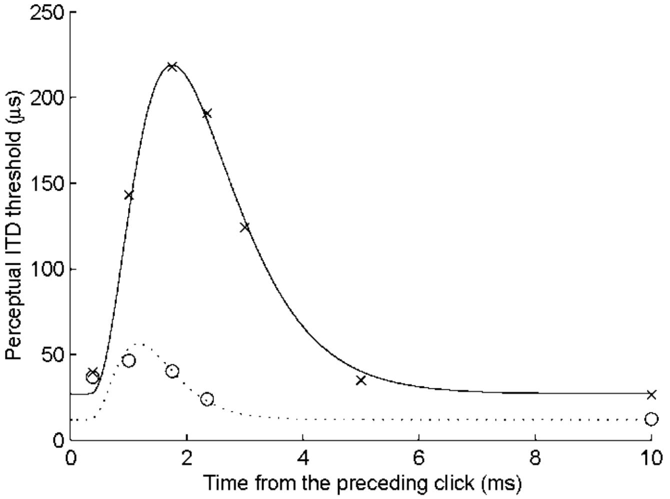


Fig. 5. The increase in the interaural time difference perceptual threshold after a preceding impulsive sound. Saberi and Perrott's experiment results are indicated by 'x' marks (experiment I) and 'o' marks (experiment II, same subjects after several extended practice sessions). The solid line is the calculated approximation of Saveri and Perrott's experiment I, and the dashed line for experiment II.

The perceptual threshold increase in the interaural time difference is quite like the post-synaptic or membrane potential of a neuron stimulated by an impulsive signal Kuffler et al. (1984). We can approximate the pattern by an exponential function $a\ \ h_i(t)\ + b$ (solid line in the figure), where

$$h_i(t) = \begin{cases} 0 & ,0 \leq t < \tau_o \\ k_i(e^{(t-\tau_o)/\tau_a} - 1)^3 e^{-(t-\tau_0)/\tau_b} & ,t \geq \tau_o \end{cases} \tag{7}$$

with $\tau_a = 1.50ms, \tau_b = 0.31ms, \tau_o = 0.3ms, k_i = 12.5$(a normalization value calculated from the peak position $t_p = -\tau_a ln(1 - 3\tau_b/)$) , $a = 192\mu s$ and $b = 27\mu s$.

This pattern is possibly a kind of impulsive aftereffect of an echo-estimation neural mechanism. However, there is a marked difference between the prospective pattern of echoes and the impulsive aftereffect. While the time extension of the prospective pattern of echoes is usually from several hundreds of milliseconds to more than 1 s depending on the reverberation time of the environment, the duration of the impulsive aftereffect is merely 5 ms.

To bridge the gap between the prospective pattern of echoes and the impulsive aftereffect derived from psychological tests, let us consider an impulsive sound and a series of its reflections as shown in Fig. 1.2. We note that the impulsive sound and its reflections will trigger a chain aftereffect so that the duration of the total aftereffect will be extended. It is because, in fact, the received sound itself includes echoes and thus can be represented as the convolution of the original sound and the impulse response of the environment. Therefore, the decay feature of the impulse response is already included in the received sound, i.e., the decay parameter in the EA model is not critical for a normal reverberant environment.

Time variance of echo estimation

Recent studies have shown that the precedence effect is active and dynamic. Blauert and Col Blauert and Col (1989, 1992), Clifton and Freyman Clifton (1987); Clifton and Freyman (1989) and Duda Duda (1996) reported that the precedence effect can break down and become re-established in an anechoic chamber or hearing test by headphones.

The echo threshold of the precedence effect changed when the subjects were presented with a series of "conditioning" clicks before tests Freyman et al. (1991) and also depends on what the subjects expect to hear Clifton et al. (1994).

The above results suggest that the precedence effect is time variant and adaptive to different environments and different sound sources. The time variance of the precedence effect is also shown in the results of Saberi and Perrott's experiments. As shown in Fig. 1.5 ('o' marks), the temporal increase of perceptual thresholds of the interaural time difference almost disappeared after some dense practice sessions by headphones. The reduced increase of the perceptual threshold can be approximated by the same exponential function Eq (7) as mentioned above, by just modifying two parameters ($\tau_b = 0.22ms, b = 12\mu s$, dashed line in Fig. 5). The decrease of the offset *b* may be due to the perceptual improvement by repeating the tests, and the change of time constant τ_b means a change in the condition of echo-estimation mechanism. However, the dynamics of the precedence effect are complex and more computational studies are required in the future to solve the problem completely.

Criteria for a transient onset

As mentioned above, the inhibition control I_c is assumed to depend on the relative strength of observed sounds and estimated echoes. We use a sigmoid like function for inhibition generation as shown in Fig. 6, where $\rho(t) = s(t)/s_e(t)$. The inhibition control will take value 1 when the sound-to-echo ratio exceeds than an open (or free-pass) threshold ρ_o, and in the opposite the inhibition control will take the value 0, when *rho*(*t*) is less than a closed (or inhibition) threshold ρ_c.

Since the echo-inhibition is depending on the relative strength of the sound-to-echo ratio, the model can explain the results of the experiments by Haas

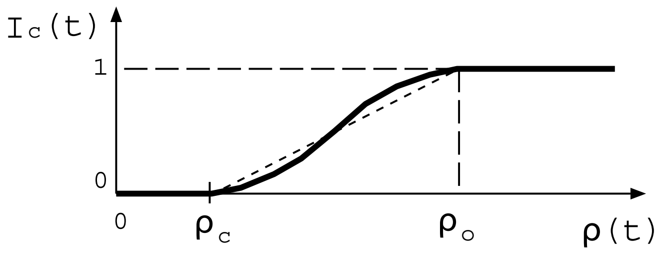


Fig. 6. The inhibition control as function of sound-to-echo ratio $\rho(t)$.

(1951) and Snow (1953) shown that the precedence effect can be canceled by a higher intensity succeeding sound.
By the EA model, we can check the availability of an onset for sound localization. We refer to a sound portion where $\rho \geq \rho_o$as an "echo-free" onset and that where $\rho \geq \rho_c$ as an available onset.
Assume that the sound level is a constant value $\mathbf{s_0}$ before t_0, and after t_0 the sound increased a level Δs for time period τ_0 (the dacay parameter in the EA model). The sound-to-echo ratio at $t = t_0 + \tau_0$ can be estimated as

$$\rho(t)\,|_{t=t_0+\tau_0} = \frac{s(t)}{\underset{t'\leq t_0+\tau_0}{Max}\ \{s(t-t')s_e(t')\}} = \frac{s(t_0)+\Delta s}{ks(t_0)} = \frac{1}{k}(1+\frac{\Delta s}{s_0}). \tag{8}$$

It shows that the sound-to-echo ratio depends on the relative increase of the sound level. This explains why a transient onset is needed for the precedence effect Rakerd and Hartmann (1985), and is also consistent with the results obtained by Franssen's experiments Franssen (1959) that a constant-level continuous sound contributions little to sound localization.

Explanation for the Haas effect
To predict the inhibition related to the results of Haas's tests, we need to analyze the "continuity" of the stimuli. Here, the continuity is evaluated by the average length of the components in individual frequencies. Since the precedence effect is considered as independent for different frequencies, a continuous articulation with a change of frequency is also considered as a discontinuity. From this meaning, we consider that the average length of the phonemes is useful for roughly estimating the continuity for speech sound.
Statistical investigations of the Japanese phonemes Kimura (1988) have shown that the five vowels in Japanese speech have an average duration of about 100 ms and the consonants have an average duration of 14 ms to over 100 ms (e.g. 't': 15 ms, 'r': 14 ms, 'sh': 146 ms and 's': 129 ms). Thus, we can roughly conclude that the continuity of Japanese speech is more than 15 ms. Regardless of the difference for different languages, we assume the speech stimuli used in Haas's tests have the same continuity.
With this knowledge in mind, we can easily understand that the onsets in the delayed speech will overlap with the continuous portions of non-delayed speech. The contribution of delayed speech to localization will thus be inhibited, as illustrated in Fig. 7(a). The overlap probability is high when the time delay is small. When the time delay becomes longer than about 14 ms **(**t_2 in Fig. 7(a)), the onsets of the delayed speech begin to neet with the next quiet portions of non-delayed speech. The inhibition will thus turn back to low according to the average ratio of silence-to-continuity portion of the speech. We can therefore expect that the inhibition of the delayed speech will have the curve as shown in Fig. 7(b). For noise stimuli, the overlap probability will decrease more quickly and remain at a relatively high level because the rate of the quiet portion is less and the average length of the continuous portion becomes shorter than speech stimuli. This tendency is more conspicuous for high frequency noise.

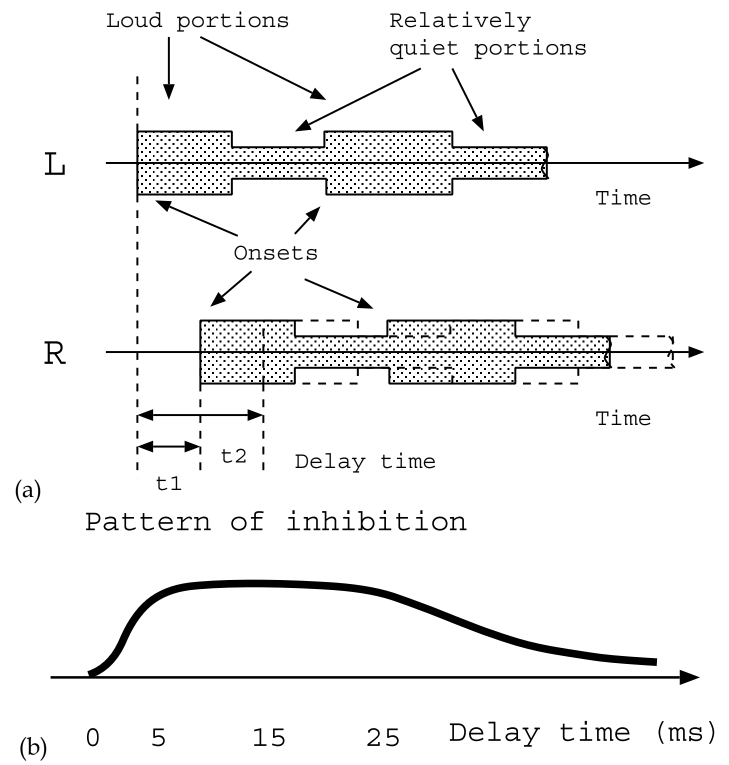


Fig. 7. (a) The \continuity" of a speech sound; (b) the pattern of inhibition.

5. Localization of multiple sound sources

We can construct a robot auditory system with four microphones arranged in the surface of a spherical head. Three of the microphones at the level of the center and one on the top of the spherical head. By the four microphones, we can solve the front-back and up-down ambiguity. To reduce the number of time difference candidates estimated from the phase differences, restrictions between the time differences of different microphone pairs were introduced. To cope with the echoes and reverberation caused by walls, ceiling, floor and other objects the EA model of the precedence effect are to calculate the weights for the ATDs. The time difference histograms are then mapped to a 2-D azimuth-elevation histogram where we can identify sound sources by its peaks Li et al. (2007).

5.1 ATD as the function of azimuth and elevation

As shown in Fig. 8, M_1, M_2, M_3 and M_4 are the four microphones. We define the positions by the spherical polar coordinates with the center of the robot head as the origin, (r (cm), θ (deg), ρ (deg)). Than, the locations of the four microphones can be described by M_1(15, 0, 90), M_2(15, 180, 0), M_3(15, 60, 0)and M_4(15, 300, 0), where the radius of the robot head is equal to 15 cm.

Consider a plane contains sound source *S*, robot head center *O* and microphone M*ᵢ* as shown in Fig 9., the distance between the sound source and each microphone can be obtained as

$$d_i = \begin{cases} |\vec{OS} - O\vec{M}_i| & (SP \geq SM_i) \\ SP + arc(PM_i) & (SP < SM_i) \end{cases} \tag{9}$$

Denoting the azimuth of sound source as θ and the elevation as ρ, and *D* as the distance from the robot head center to the sound source, we have

$$\vec{OS} = D(cos\rho cos\theta, sin\theta, cos\rho sin\rho) \tag{10}$$

$$O\vec{M}_i = R(cos\rho M_i cos\theta M_i, sin\rho M_i, cos\rho M_i sin\theta M_i) \tag{11}$$

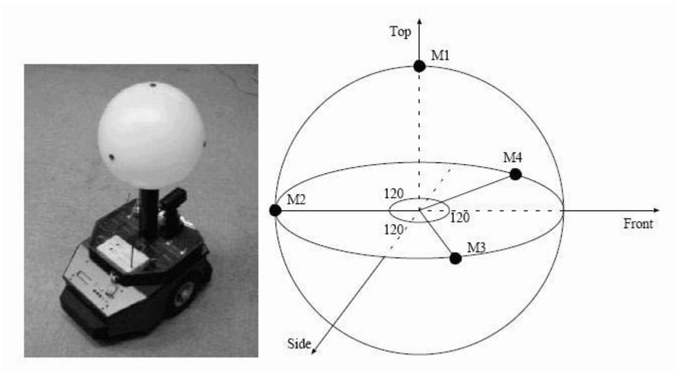


Fig. 8. The robot system and a spherical head with a four-microphone set.

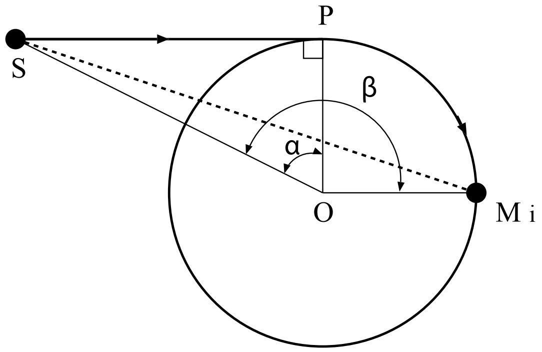


Fig. 9. The distance of sound path from the source to each microphone.

where ***R*** is radius of the robot head, and ρ_{M_i} are the azimuth and elevation of microphone ***i*.** The ATD between two microphones mainly depends on the azimuth and elevation of sound source. When *D* is very larger than $R(D \gg R)$, the influence caused by the difference of *D* can be ignored. Thus, the arrival time differences can be denoted as the function of θ and ρ

as

$$\Delta t_{ij}(\theta\rho) = \frac{d_i(\theta,\rho) - d_j(\theta,\rho)}{v} \tag{12}$$

where v is the sound velocity and $i, j = 1, 2, 3, 4\ i \neq j$.
Fig 10 shows the calculation results of ATDs between microphone M_1 and M_2.

5.2 Restrictions between ATD candidates

Because of the phase wrapping of high frequency components, the ATDs can not be determined uniquely. We obtain the ATD candidates from the phase

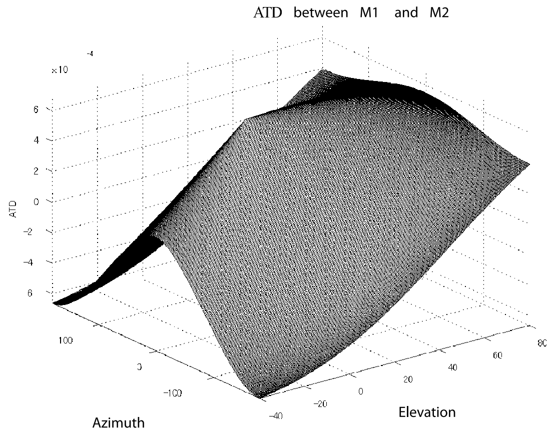


Fig. 10. ATDs between SM_1 and SM_2

differences of each frequency band for each microphone pair,

$$\Delta z_{ij}^{(k)} = \frac{\Delta\psi_{ij} + 2\pi n}{2\pi f_k} \tag{13}$$

where $\Delta\psi_{ij}$ is the measured phase difference between a microphone pair *i* and *j* in the fcth frequency band with center frequency f_k, and n is an arbitrary integer limited by $|\Delta t_{ij}^{(k)}| \leq \Delta T = \frac{2}{3}\pi R/C$.

To reduce the number of ATD candidates, we consider the sum ATDs which forms a circulation start from one microphone (ex, microphone 1), through the other three microphones and finally finish at the same microphone. The sum of ATDs in such a circulation should be zero. We have six different such paths, and if we ignore the difference in direction we can have the following three restrictions.

Zero summation restrictions:

$$\begin{aligned}\Delta t_{12}^{(k)} + \Delta t_{23}^{(k)} + \Delta t_{34}^{(k)} + \Delta t_{41}^{(k)} &= 0\\ \Delta t_{13}^{(k)} + \Delta t_{34}^{(k)} + \Delta t_{42}^{(k)} + \Delta t_{21}^{(k)} &= 0\\ \Delta t_{14}^{(k)} + \Delta t_{42}^{(k)} + \Delta t_{23}^{(k)} + \Delta t_{31}^{(k)} &= 0\end{aligned} \tag{14}$$

Similarly, we can have the restrictions for the absolute sum between the circulation ATDs

$$\begin{aligned}T_{min} \le |\Delta t_{12}^{(k)}| + |\Delta t_{23}^{(k)}| + |\Delta t_{34}^{(k)}| + |\Delta t_{41}^{(k)}| \le T_{max}\\ T_{min} \le |\Delta t_{13}^{(k)}| + |\Delta t_{34}^{(k)}| + |\Delta t_{42}^{(k)}| + |\Delta t_{21}^{(k)}| \le T_{max}\\ T_{min} \le |\Delta t_{14}^{(k)}| + |\Delta t_{42}^{(k)}| + |\Delta t_{23}^{(k)}| + |\Delta t_{31}^{(k)}| \le T_{max}\end{aligned} \tag{15}$$

where $T_{min} = 2R/C$ for sound from top direction with 90 degrees elevation, and $T_{max} = (\frac{4}{3}\pi + \frac{3}{2})R/C$ for sound from the back direction with 180 degrees azimuth and 0 degree elevation.

For sound sources in the horizontal plane, the restrictions are reduced to the following forms,

$$\Delta t_{12}^{(k)} + \Delta t_{23}^{(k)} + \Delta t_{31}^{(k)} = 0 \tag{16}$$

and

$$T'_{min} \le |\Delta t_{12}^{(k)}| + |\Delta t_{23}^{(k)}| + |\Delta t_{31}^{(k)}| \le T'_{max}$$

where $T'_{min} = (2 + \frac{2}{3}\pi)R$ and $T'_{max} = \frac{4}{3}\pi R$.

5.3 Integration and mapping of ATD histograms

By summarizing all of the ATD candidates of different frequency bands for microphone pair *i* and *j* with the weights calculated by the EA model, we can form an histogram $h_{ij}(\Delta t_{ij})$. Since $\Delta t_{ij}$is a function of θ and ρ, we can denote the histogram as $h_{ij}(\Delta t_{ij}(\theta, \rho))$. This histogram, after a smoothing operation, can be a continuous function for Δt_{ij} and thus for θ and ρ. We now consider a new function $p_{ij}(\theta, \rho)$ of θ and ρ, and let

$$p_{ij}(\theta, \rho) = h_{ij}(\Delta t_{ij}(\theta, \rho)) \tag{17}$$

Here, $p_{ij}(\theta, \rho)$ is a two dimensional histogram, differing with h_{ij} which is an one dimensional histogram of Δt_{ij}. We call this transformation as a 'mapping' from ATD histogram to azimuth-elevation histogram, a one-to-many mapping.

The azimuth-elevation histogram p_{ij} of a single microphone pair is not enough to determine the θ and ρ of sound sources. The peaks in the azimuth-elevation histogram provide a necessary condition only but not sufficient. The sufficient condition can be integrated by taking the geometric average for different microphone pairs,

$$P(\theta, \rho) = \left[\prod_{i \ne j} p_{ij}(\theta, \rho)\right]^{1/6} \tag{18}$$

An horizontal version of this mapping is shown in Figure 11.
The final histogram $P(\theta, \rho)$, after normalization, can be consider as a possibility function for spatial distribution of sound source in all of the directions (θ, ρ).

5.4 Experiments and Results

The experiments were carried out in both an anechoic chamber and an ordinary room. The anechoic chamber has a size of 5.5 $\times$ 5.5 $\times$ 5.5 **m³,** and the ordinary room is of size 4 $\times$ 6 $\times$ 3 **m³,** without any acoustic treatment for its floor, ceiling and walls. The robot head was set on a height, 1 m above the floor.

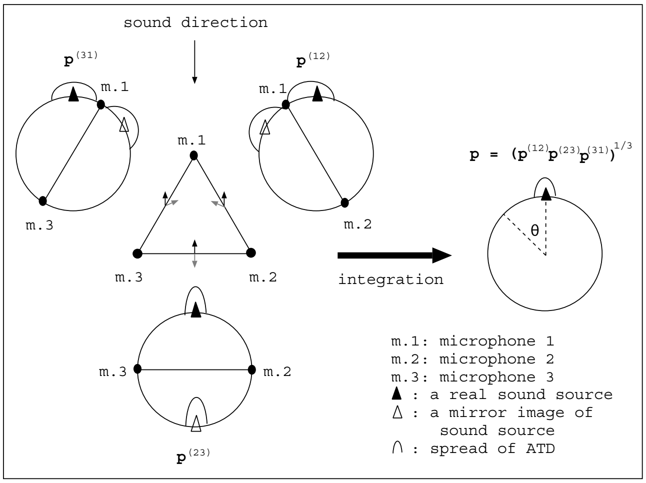


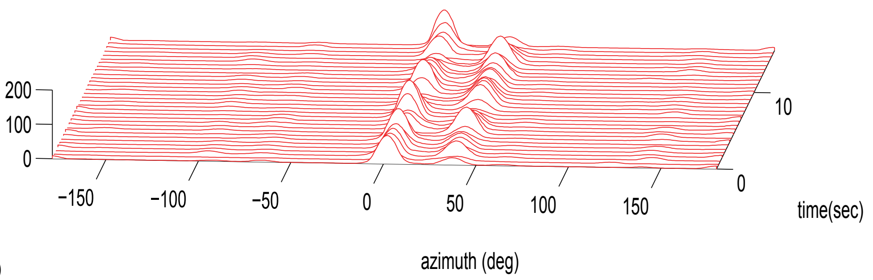
Fig. 11. Mapping and integration of ATDs to the azimuth histogram.

Localization of horizontal sound sources

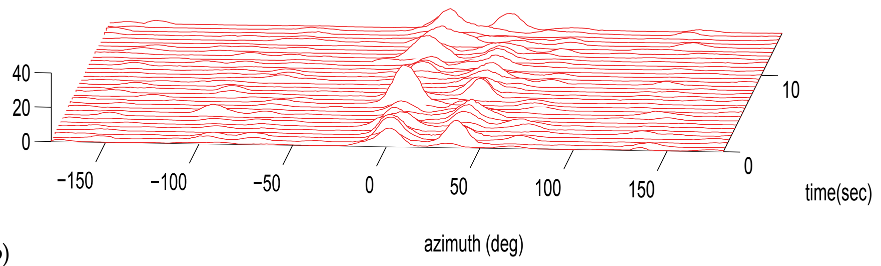
Sound source (1) was a recording of a radio weather forecast presented by a male announcer (speaker 1 at 0 degrees of azimuth), and sound source (2) was a recording of a fast radio talk show with one male and one female host (speaker 2 at 38 degrees azimuth). The overlap portion of Sound source (1) and sound source (2) is about 16 seconds.
The resulting azimuth histograms are shown in Figure 12. The time segment for each azimuth histogram is 0.5 s.
The peaks appear clearly around 0 degree and 38 degrees over all time segments in the histograms of the anechoic chamber. The positions of major peaks are in the regions of [0,4] and [35,39] degrees, *i.e.,* the sound source 1 was localized in 2(±2) degrees and sound source 2 in 37(±2) degrees. The maximum absolute error is 4 degrees (regardless of the setup errors).
The histograms of the normal room, however, show more disorder comparing to the anechoic chamber. The scores are smaller and the size of the peaks is not consistent. We smoothed the histograms by a two-dimension gaussian function ($\sigma_t = 1$ second and $\sigma_\theta = 2$degrees) as shown in Figure 12.(c). In the smoothed histogram, the time resolution decreased to about 2 seconds, but the peak positions became more consistent.
The positions of major peaks are in the regions of [-2,2] and [33,37] degrees, *i.e.,* the sound

source 1 was localized in 0(±2) degrees and sound source 2 in 35(±2) degrees.

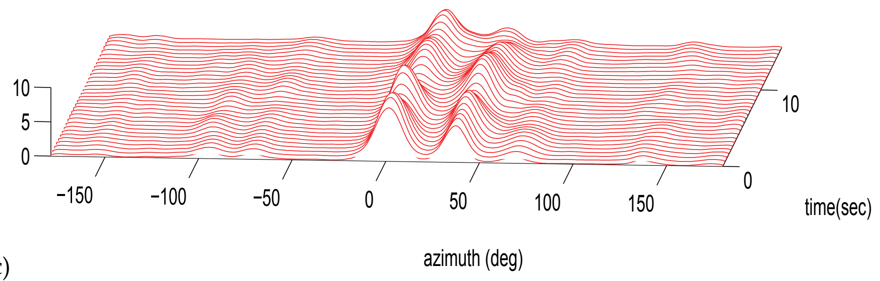
Localization of spatial sound sources
We used two loudspeakers to generate two different sound sources concurrently at different locations with a intensity difference.



(a)



(b)



(c)

Fig. 12. Azimuth histograms obtained in the anechoic chamber (a) and the normal room (b), with time segments of 0.5 second and total length of 16 seconds. Figure (c) is the smoothed azimuth histograms in the normal room, by two-dimension gaussian function ($\sigma_t = 1$ second and $\sigma_\theta = 2$ degrees).

The signals received by the four microphones were divided into frames of 250 ms length. The azimuth-elevation histogram was calculated for each time frame. To reduce noise and increase accuracy, histograms of every 10 frames were summed and applied with a threshold.
Fig. 13 shows a resulting azimuth-elevation histogram for sound sources at S_1 (0°, 0°) and S_2(−90°, 0°) in the anechoic chamber.

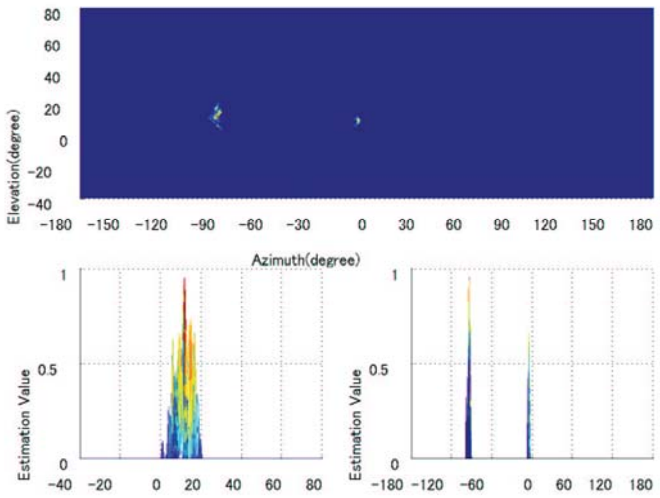


Fig. 13. A resulting average azimuth-elevation histogram for 10 time frames for sound source S1(0°, 0°) and S2(−90°, 0°) in an anechoic chamber.

Fig. 14 shows the resulting azimuth-elevation histogram for sound sources in the ordinary room. From the figure, we can identify the two sound sources by the peaks clearly.
The average localization errors by the peak positions of multiple sound sources at different locations in the ordinary room are shown in Table. 1.1.

Sound sources direction	S1(0o, 0o) S2(-90o, 0o)	S1(0o, 15o) S2(90o, 15o)	S1(0o, 15o) S2(90o,3 0o)
Average peaks direction errors	(7o, 8o) (4o, 12o)	(4o, 2o) (4o, 2o)	(2o, 6o) (6o, 8o)

Table 1. An average azimuth-elevation histogram for 10 time frames of two sound sources in an ordinary.

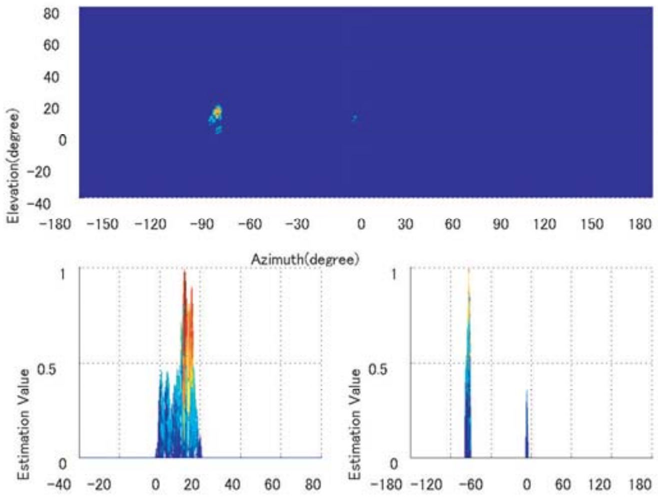


Fig. 14. A resulting average azimuth-elevation histogram for 10 time frames for sound source $S_1(0°, 0°)$ and $S_2(-90°, 0°)$ in an ordinary room.

6 Navigating a robot to a sound source

6.1 System

In this paper, we describe a mobile robot equipped with a simplified real time sound localization system which involves the function to cope with echoes and reverberations. By audition, a robot can find an object behind obstacles and even outside of a room by tracing back the sound path from the source Huang et al. (1999, 1997a). Experiments of robot navigation by the sound localization system have been conducted in various setups and conditions to demonstrate the effectiveness of the system.

As shown in Figure 15, the three omnidirectional microphones are placed on the circumference of a circle in a manner that the three microphones form a regular triangle on a horizontal plane, where the radius of the circle was variable. The mobile robot (RWI B-12) is wheel-based and equipped with a sonar system for obstacle detection and a CCD camera system reserved for visual processing.

To make it easy to implement the method in a mobile robot, the system uses only a single frequency band (central frequency 1kHz, band width 600 Hz). The frequency band and its width is chosen so that the maximum phase difference between two microphones will be less than π. Therefore the time differences can be uniquely determined from the zero-crossing points of wave forms.

The steering unit of the system has four modes,

• Localization mode: localizing sound sources

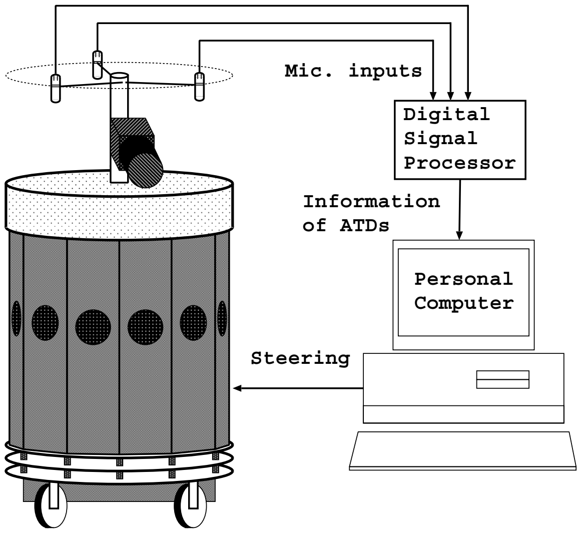


Fig. 15. A robot equipped with ears (microphones)

- Forward movement mode: going straightforward
- Avoidance mode: avoiding obstacles
- Stop mode: waiting for instructions.

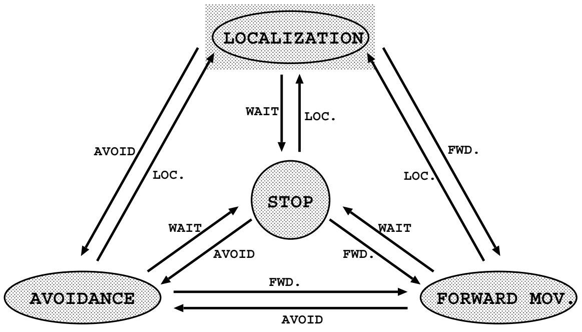


Fig. 16. Modes for robot steering

Transfers among the four modes are controlled by the commands from the personal computer: "localize sound", "avoid obstacle", "go forward" and "wait", as shown in Figure 16. The method of obstacle avoidance is simply implemented by turning the direction of the robot clockwise whenever an obstacle is detected in front of it by the sonar system.

6.2 Experiments

The experiments were conducted in an ordinary room, where background noise of up to 40 dB is generated by computer fans, an air conditioner and other environment noise coming in through the windows. The sound sources used were a 1 kHz brief sinusoidal sound presented by a loudspeaker and hand-clapping sounds.

Obstacle Avoidance

In this experiment, an obstacle was located between the initial position of the robot and the sound source of goal (a speaker emitting intermittent 1kHz sounds) as shown in Figure 1.17. The obstacle was about 0.47m high, lower than the mount position of the microphone set, so that the direct sound path from the sound source to the robot was not obstructed. The result shows that the mobile robot could correctly localize the sound source and move toward the sound source position avoiding the obstacle in between. Here, the source direction was provided by the sound localization unit and the obstacle avoidance was achieved by the sonar system.

If the obstacle is big enough and higher than the position of microphones and sound source, there will be no direct path from sound source to microphones. In this case, the sound localization unit will localize the sound source by the minimum path of sound-to-microphone (see next section). In generally, it means the system will localize the sound source at the edge of the obstacle (the turning point of the minimum path). The edge position of the obstacle is helpful, since after the robot approached the edge the robot can localize the sound source again and then finally approach the source.

Approaching an Invisible Sound Source

In this experiment, the sound source was invisible to the robot and the direct path from the sound source to the robot was blocked. The robot was initially positioned outside a room, while the sound source was positioned inside (Figure 18). In this situation, instead of the direct sound, the sound from the minimum path plays an important role. The sound direction perceived by the robot will depend on the last turning position of the minimum path from the sound source to the robot.

The results of estimation in positions (X1 - X6) are shown in Table 2. From the results, the robot did tend to localize the sound at the last turning position of the minimum path, the edge of the door. The localization accuracy was lower than what was obtained with a direct path, especially when the robot was far from the sound source. This is because the narrow corridor can act as a large guiding tube, where reflections will be funneled in the direction of the corridor to form a mixed sound with expanded arrival directions. However, the localized sound direction always pointed to the same side of the corridor, and after some approaches and corrections the robot was able to find the turning position and finally localize the sound source inside the room.

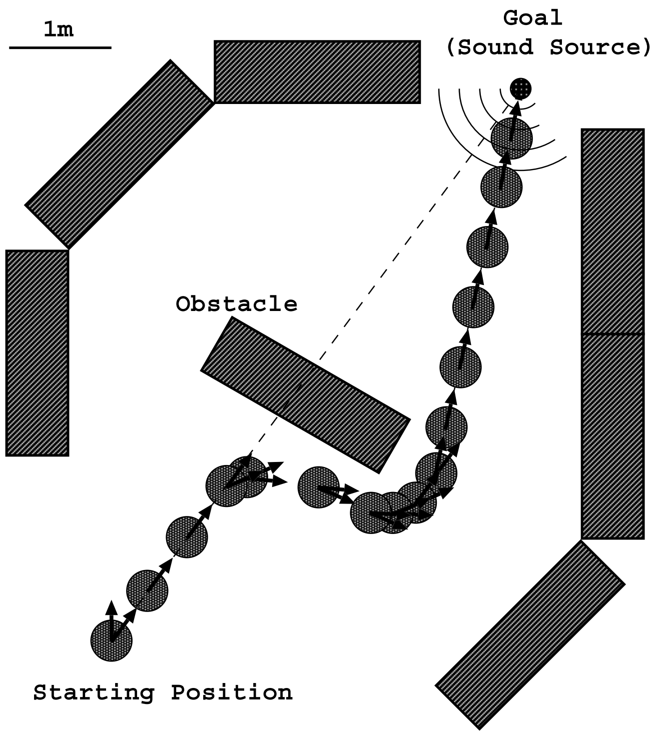


Fig. 17. Obstacle avoidance

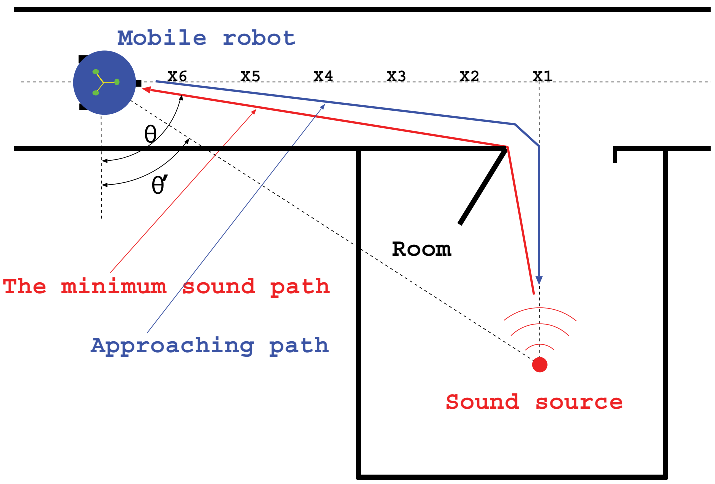


Fig. 18. Approaching a invisible sound source

measurement	X1	X2	X3	X4	X5	X6
1	0	49	60	75	*	62
2	1	41	*	79	76	81
3	0	46	64	76	80	87
4	1	39	65	75	*	*
5	0	43	67	81	85	*
6	1	44	69	71	83	68
7	1	44	67	82	81	50
8	1	45	66	79	*	81
9	0	45	55	76	78	75
10	1	46	61	78	81	47
average	0.6	44.2	63.8	77.2	80.6	68.9
θ'	0	23.1	40.5	52.0	59.7	64.9
θ	0	44.5	66.7	74.7	78.8	81.0

*: failure of localization

Table 2. Azimuth estimation of a invisible sound source

7. References

Aarabi, P. and Zaky, S. (2000). Integrated vision and sound localization. In *Proc. 3rd Int. Conf. Information Fusion,* Paris.

Bekey, G. A. (2005). *Audonomous Robots: From Biological Inspiration to Implementation and Control (Intellegent Robotics and Autonomous Agents).* MIT Pr.

Blauert, J. (1997). *Spatial hearing: the psychophysics of human sound localization.* The MIT Press, London, revised edition.

Blauert, J. and Col, J. P. (1989). Etude de quelques aspects temporels de l'audetion spatiale (A study of certain temporal effects of spatial hearing). Note-laboratoire LMA 118, CRNS, Marseille.

Blauert, J. and Col, J. P. (1992). A study of temporal effect in spatial hearing. In Cazals, Demany, and Horner, editors, *Auditory Psychology and Perception,* pages 531-538. Pergamon Press.

Bodden, M. (1993). Modeling human sound-source localization and the cocktail-party-effect. *Acta Acustica,* 1:43-55.

Bregman, A. S. (1990). *Auditory Scene Analysis: The Perceptual Organization ofSound.* The MIT Press, London.

Cherry, E. C. (1953). Some experiments on the recognition of speech with one and with two ears. *J. Acoust. Soc. Am.,* 25:975-979.

Clifton, R. K. (1987). Breakdown of echo suppression in the precedence effect. *J. Acoust. Soc. Am.,* 82:1834-1835.

Clifton, R. K. and Freyman, R. L. (1989). Effect of click rate and delay and breakdown of the precedence effect. *Percept. Psychophys.,* 46:139-145.

Clifton, R. K., Freyman, R. L., Litovsky, R. Y., and McCall, D. (1994). Listern-ers' expectations about echoes can raise or lower echo threshold. *J. Acoust. Soc. Am.,* 95:1525-1533.

Clifton, R. K., Morrongiello, B. A., and Dowd, J. M. (1984). A developmental look at an auditory illusion: The precedence effect. *Dev. Psychobiol.,* 17:519536.

Cooke, M. (1993). *Modeling Auditory Processing and Organisation.* Cambridge University Press, Cambridge.

Duda, R. O. (1996). Auditory localization demonstrations. *Acustica,* 82:346355.

Ellis, D. P. W. (1994). A computer model of psychoacoustic grouping rules. In *Proc. 12th Int. Conf. on Pattern Recognition.*

Franssen, N. V. (1959). Eigenschaften des naturlichen Richtungshorens und ihre Anwendung auf die Stereophonie (The properties of natural directional hearing and their application to stereophony). In *Proc. 3rd Int. Congr. Acoustics,* volume 1, pages 787-790.

Freyman, R. L., Clifton, R. K., and Litovsky, R. Y. (1991). Dynamic processes in the precedence effect. *J. Acoust. Soc. Am.,* 90:874-884.

Gardner, M. B. (1968). Historical background of the Haas and/or precedence effect. *J. Acoust. Soc. Am.,* 43:1243-1248.

Gelfand, S. A. (1998). *Hearing — An Introduction to Psychological and Physiological Acoustics — (Third Edition, Revised, and Expanded,).* Marcel Dekker, Inc., New York.

Haas, H. (1951). Uber den eingluss eines einfachechos auf die horsamkeit von sprache. *Acustica,* 1:49-58. English translation in: "The influence of a single echo on the audibility of speech", *J. Audio Eng. Soc.,* Vol. 20, pp. 146-159, (1972).

Hafter, E. R., Buell, T. N., and Richards, V. M. (1988). Onset-coding in later-alization: Its form, site, and function. In Edelman, G. M., Gall, W. E., and Cowan, W. M., editors, *Auditory Function: Neurobiological Bases of Hearing,* pages 647-676. Wiley and Sons, New York.

Harris, G. G., Flanagan, J. L., and Watson, B. J. (1963). Binaural interaction of a click with a click pair. *J. Acoust. Soc. Am.,* 35:672-678.

Hartmann, W. M. and Rakerd, B. (1989). Localization of sound in room IV: The Franssen effect. *J. Acoust. Soc. Am.,* 86:1366-1373.

Heffner, R. S. and Heffner, H. E. (1992). Evolution of sound localization in mammals. In D. B. Webster, R. R. F. and Popper, A. N., editors, *The evolutionary biology ofhearing,* pages 691-715. Springer-Verlag, New York.

Hornstein, J., Lopes, M., Statos-Victor, J., and Lacerda, F. (2006). Sound localization for humanoid robot - building audio-motor maps based on the hrtf. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems.* IEEE/RSJ.

Huang, J., Ohnishi, N., and Sugie, N. (1995). A biomimetic system for localization and separation of multiple sound sources. *IEEE Trans. Instrum. and Meas.,* 44(3):733-738.

Huang, J., Ohnishi, N., and Sugie, N. (1997a). Building ears for robots: Sound localization and separation. *Artificial Life and Robotics,* 1(4):157-163.

Huang, J., Ohnishi, N., and Sugie, N. (1997b). Sound localization in reverberant environment based on the model of the precedence effect. *IEEE Trans. Instrum. and Meas.,* 46(4):842-846.

Huang, J., Supaongprapa, T., Terakura, I., Wang, F., Ohnishi, N., and Sugie, N. (1999). A model based sound localization system and its application to robot navigation. *Robotics and Autonomous Systems,* 27(4):199-209.

Huang, J., Zhao, C., Ohtake, Y., Li, H., and Zhao, Q. (2006). Robot position identification using specially designed landmarks. In *Proc. Instrum. Meas. Technol. Conf.,* Sorrento. IEEE.

Johnson, D. H. and Dudgeon, D. E. (1993). *Array Signal Processing: Concepts and Techniques.* PTR Prentice-Hall, NJ.

Kimura, S. (1988). Investigation on variations of acoustic segment duration. No. 1-2-14, Proc. Spring Meet. Acoust. Soc. Jpn. (Japanese).

Knudsen, E. I. (1981). The hearing of the barn owl. *Sci. Am.,* 245(6):82-91.

Konishi, M. (1986). Centrally synthesized maps of sensory space. *Trends in Neuroscience,* 9(4):163-168.

Kuffler, S. W., Nicholls, J. G., and Martin, A. R. (1984). *From Neuron to Brain: A Cellular Approach to the Function of the Nervous System.* Sinauer Associates Inc., Sunderland, MA, second edition.

Lehn, K. H. (1997). Modeling binaural auditory scene analysis by a temporal fuzzy cluster analysis approach. In *Proc. IEEE Workshop WASPAA'97.*

Li, H., Yoshiara, T., Zhao, Q., Watanabe, T., and Huang, J. (2007). A spatial sound localization system for mobile robots. In *Proc. Instrum. Meas. Technol. Conf.,* Warsaw. IEEE.

Lindemann, W. (1986a). Extension of a binaural cross-correlation model by contralateral inhibition. i. simulation of lateralization for stationary signals. *J. Acoust. Soc. Am.,* 80:1608-1622.

Lindemann, W. (1986b). Extension of a binaural cross-correlation model by contralateral inhibition. ii. the low of the first wavefront. *J. Acoust. Soc. Am.,* 80:1623-1630.

Litovsky, R. Y. and Macmillan, N. A. (1994). Sound localization precision under conditions of the precedence effect: Effects of azimuth and standard stimuli. *J. Acoust. Soc. Am.,* 96:752-758.

Martin, K. D. (1997). Echo suppression in a computational model of the precedence effect. In *Proc. IEEE Workshop WASPAA'97.*

McFadden, D. (1973). Precedence effects and auditory cells with long characteristic delays. *J. Acoust. Soc. Am.,* 54:528-530.

Medioni, G. and Kang, S. B., editors (2005). *Emerging Topics in Computer Vision.* PTR Prentice-Hall.

Nishizawa, Y., Yagi, Y., and Yachida, M. (1993). Generation of environmental map and estimation of free space for a mobile robot using omnidirectional image sensor COPIS. *J. Robotics Soc. Japan,* 11(6):868-874. (Japanese).

Oertel, D. and Wickesberg, R. E. (1996). A mechanism for the suppression of echoes in the cocklear nuclei. In Ainsworth, W. A., editor, *Advances in Speech, Hearing and Language Processing,* volume 3 (Part B), pages 293-321. JAI Press Inc., London.

Okuno, H. G., Nakadai, K., Hidai, K., Mizoguchi, H., and Kitano, H. (2001). Human-robot interaction through real-time auditory and visual multiple-talker tracking. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems,*
pages 1402-1409. IEEE/RSJ.

Parkin, P. H. and Humphreys, H. R. (1958). *Acoustics, Noise and Buildings.* Faber & Faber, London.

Rakerd, B. and Hartmann, W. M. (1985). Localization of sound in rooms, II: The effects of a single reflecting surface. *J. Acoust. Soc. Am.,* 78:524-533.

Rakerd, B. and Hartmann, W. M. (1992). Precedence effect with and without interaural differences — Sound localization in three planes. *J. Acoust. Soc.*
Am., 92:2296(A).

Saberi, K. and Perrott, D. R. (1990). Lateralization thresholds obtained under conditions in which the precedence effect is assumed to operate. *J. Acoust. Soc. Am.,* 87:1732-1737.

Shen, J. X. (1993). A peripheral mechanism for auditory directionality in the bushcricket gampsocleis gratiosa: Acoustic tracheal system. *J. Acoust. Soc. Am.,* 94:1211-1217.

Snow, W. B. (1953). Basic principles of stereophonic sound. *J. Soc. Motion Pict. Telev. Eng.,* 61:567-587.

Takahashi, T. and Konishi, M. (1986). Selectivity for interaural time difference in the owl's midbrain. *J. Neuroscience,* 6(12):3413-3422.

Thurlow, W. R., Marten, A. E., and Bhatt, B. J. (1965). Localization aftereffects with pulse-tone and pulse-pulse stimuli. *J. Acoust. Soc. Am.* , 37:837-842.

Valin, J. M., Michaud, F., and Rouat, J. (2007). Robust localization and tracking of simultaneous moving sound sources using beamforming and particle filtering. *Robotics and Autonomous Systems,* 55(3):216-228.

von Bekesy, G. (1960). *Experiments in Hearing (pp.288-301,609-634).* McGraw Hill, New York. Translated from: Zur Theorie des Horens: Uber das Richtung-shoren bei einer Zeitdifferenz oder Lautstarkenungleichheit der beiderseitigen Schalleinwirkungen, *Physik. Z.,* **31,** pp.824-835, 857-868, (1930).

Wallach, H., Newman, E. B., and Rosenzweig, M. R. (1949). The precedence effect in sound localization. *J. Psychol. Am.,* 62(3):315-336.

Zurek, P. M. (1980). The precedence effect and its possible role in the avoidance of interaural ambiguities. *J. Acoust. Soc. Am.* , 67:952-964.

Zurek, P. M. (1987). The precedence effect. In Yost, W. A. and Gourevitch, G., editors, *Directional hearing,* pages 85-105. Springer-Verlag, New York.

Objects Localization and Differentiation Using Ultrasonic Sensors

Bogdan Kreczmer
*The Institute of Computer Engineering,
Control and Robotics, University of Technology
Poland*

1. Introduction

The most crucial problem for the mobile robot navigation is obstacles detection and their localization. The determination of obstacle position should be as accurate as possible in order to support robot self-localization procedures. In order to increase its efficiency the recognition of some feature of obstacles shapes should be done.
The most advanced systems use laser range-finder and vision to solve this task. They allow to obtain a lot of data of a robot environment and the delivered information is quite precise. Unfortunately these devices have some important drawbacks. Laser scanning range-finders are still expensive. Their another drawback is that they scan only in a single plain. It causes that some obstacle cannot be detected. There are also available 3D laser range-finders. But measurements performed by them are time consuming. Therefore it is rather difficult to use them for on-line mobile robot navigation when a robot moves during a measurement execution. Considering vision systems the main disadvantages are computation consuming methods and a price of the system.
In this sense ultrasonic range-finders seems still to be very useful equipment for a mobile robot. Their important advantage is low price and simplicity. Considering robotics applications they seem to be very attractive comparing especially with laser range-finders and vision systems. But in current mobile robots the ultrasonic sensors are rather used as an auxiliary equipment allowing to obtain rough information of the environment. The main reason of this situation is that the obtained data from commercial ultrasonic range-finders are very difficult to interprete. In this chapter two methods are presented which makes possible to overcome some difficulties combined with ultrasonic sensing. The first one is dedicated to the problem of object differentiation. The second method addresses the problem of an object localization and simplification of necessary computations.

2. Ultrasonic sensing

The result of a measurement performed by a commercial ultrasonic range-finder is time of flight (TOF) during which an ultrasonic signal is propagated from a sender to an obstacle and, after being reflected, back to a receiver. This is enough to compute the distance of the path. In this way the form of the data obtained form sonars is very simple. Unfortunately this type of information is not easy to interpret. The main reason of their disadvantage is a wide beam of an emitted signal ($20^\circ \sim 50^\circ$). Traditional range-finder contains a single sender and a single receiver or a transducer which can work as a sender and than as a receiver. The wide emitted beam causes that they suffer from a very pure resolution. This type of beam smears the location of the object reflecting an echo and produces arcs when a rotational scan of the environment is performed. The extent of the arcs is related to the reflecting strength of the object Kuc & Siegel, (1987)Leonard & Durrant-Whyte, (1991)Kleeman & Kuc, (1995). In this case, the distance information that sonars provide is fairly accurate in depth, but not in an angle.

Another reason is combined with a frequency of an emitted wave packet. Because the frequency is usually $40kHz$ (piezo electric transducers) or $50kHz$ (electrostatic transducers) the length of the wave in air is about $8.5mm$ and $6.8mm$ respectively. When irregularities of an object surface are much smaller than the wave length of an ultrasonic signal then the surface can be treated as a kind of an acoustic mirror. Thus many objects in indoor environments can be assumed to be specular reflectors for ultrasonic waves. It causes that sometimes a sonar receives a multi-reflected echo instead of the first one. These reflections produce artifacts which are a false image of no existing object behind a real one or sometimes even in front of it. This last phenomena can be observed when several successive measurement are performed in regular short intervals. In this case it can happen that instead of receiving echo caused by a current emitted signal an echo produced by previous emission is detected. Considering cases of false images created by double reflections the well know example is a room corner. While a single sonar scanner is used, it isn't possible to distinguish it from a single wall because the same sequence of echos is obtained.

Using a single sonar it isn't possible correctly to distinguish the case of a single and multi-reflection. The picturesque description of the problem has been presented in Brown, (1985). Brown compared sensing with ultrasonics to trying to navigate in a house of mirrors using only a flash light. This is true if rooms, through which the robot navigates, contain only plain walls (in relation to the length of ultrasonic wave). Fortunately in indoor environment there are a lot of objects which are the source of direct echos. It means signals which are reflected by a single object and then detected by a sonar. But it doesn't help very much when a scanning range-finder consisting of a single sonar is used. Obtained data cannot be properly interpreted because it doesn't exist well defined one-to-one mapping between a contour of ultrasonic distance map and surfaces of objects or objects them self. In spite of that ultrasonic sensing has an immense potential to mobile robot navigation. In the animal world the well known examples of successful usage of ultrasonic waves for navigation are bats and dolphins. They can properly navigate in a very difficult conditions. For example small bats are able to fly at full speed through wire grid structures that are only slightly larger than their wingspan Cheeke, (2002). The main difference between a scanning ultrasonic range-finder and a bat is that the bat has two ears. They allow the bat to determine direction from which echo comes. In addition it was shown in Schillebeeckx et al., (2008) that a pinna can significantly influence on directivity pattern of a receiver which can be helpful for localization purposes.

But even using a single sonar it is possible to increase credibility of obtained data. It can be noticed that the most artifacts (but not all) are sources of weak echos. Kuc proposed a method to eliminate them using a standard Polaroid sonar. The 6500 ranging module controlling a Polaroid sonar can detect echoes beyond the initial one by resetting the detection circuit. The device specification suggests inserting a delay before resetting to prevent the current echo

from retriggering the detection circuit. Ignoring this suggestion, Kuc applied another method. He repeatedly reset the module immediately after each detection to generate a dense sequence of detection times Kuc, (2001).
Another approach to artifacts elimination is based on the assumption that multi-reflected echos usually comes from the direction being far from the acoustic axis of the sender. This assumption is based on the observation that that the emitted signal in such a direction is weaker comparing with the signal propagated in the direction of the acoustic axis. In consequence it cannot be expected that a strong echo will come from these directions. To determine the direction of echo arrival a binaural sonar system is needed which contains a receiver and a transducer working as a sender and a receiver. Kreczmer, (1998). However more efficient

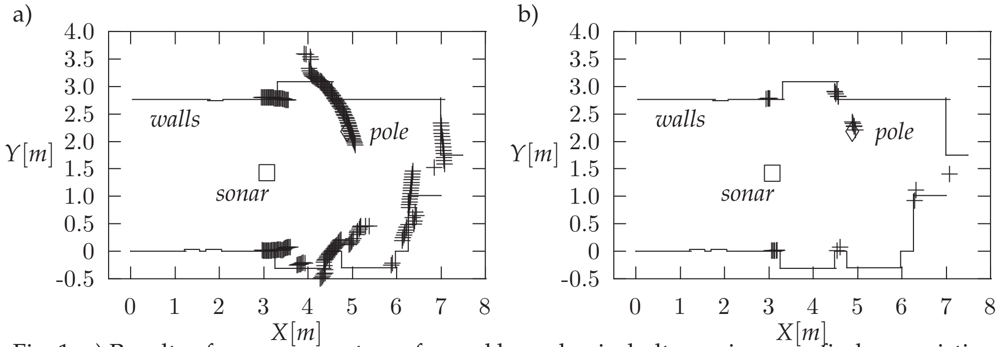


Fig. 1. a) Results of measurements performed by a classical ultrasonic range-finder consisting of a single sonar. The square marks the position of the range-finder. b) Results of measurements performed by a tri-aular sonar system

solution is a tri-aular sonar system which works as a double binaural system (two receivers and a single transducer working as a sender and receiver) The result obtained from the second pair of sonars can be used as a confirmation result obtained from the first one. It allows to reject some week echos. This solution combining with restriction to echos coming from direction being close to the acoustic axis of the system creates an efficient filter. It makes possible significantly to increase credibility of obtained data. The example of a such case is presented in fig. 1. The data obtained from an ultrasonic range-finder consisting of a single sonar are shown in fig. 1a. The range-finder scanned the surrounding at every 0.9°. It has

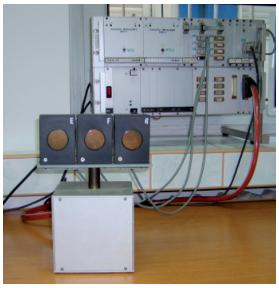


Fig. 2. The tri-aular sonar system

been built using a standard Polaroid transducer 600–series model and a ranging module series 6500. Measurements for the same scene have been performed using a tri-aural system (see fig. 2). The obtained results are presented in fig. 1b. The area around the acoustic axis of the system has been restricted up to $\pm 4^\circ$. It allowed successfully to reject most of false reading. Unfortunately some correct echos have been also rejected due to error measurements. The construction of the tri-aular sonar system has been also based on Polaroid transducers 600–series model and the 6500 ranging modules. These modules stimulate the transducer by series of 16 pulses in order to force it to emit ultrasonic signal. Then after $440\mu s$ it can be switched into a receiver mode. The module doesn't allow to start receiving without sending a signal. It is done due to necessary polarization (about $200V$) which has to be set to the electro static transducer. It is needed when the transducers works as an receiver also as a sender. To obtain such a polarization, initial pulses generated during sending mode are used for pumping electric charge. Because in the tri-aular sonar system two side sonars must work only as receivers therefore these modules were modified.

Using multi-sonar system not only some artifacts can be rejected but first of all objects can be much more precisely localized. Moreover some objects can be distinguished. There are three the most elementary reflectors: a wall, a 90° concave corner and a convex edge. In Peremans et al., (1993) it was presented a method which makes possible to localize and classify an edge and a wall. A corner and a wall are indistinguishable in this approach. The method is based on measurements of TOF. In the aforementioned paper a tri-aular sonar system was proposed to solve the problem of the object localization and classification.

An other approach was proposed in Kleeman & Kuc, (1995). In this approach a sonar system which consists of three ultrasonic transducers is also used and TOF is measured. But they are in different way arrange. Additionally two of them are used as transmitters and all of them are used as receivers. Kleeman and Kuc showed that to distinguish wall, edge and corner, measurements performed by using at least two transmitters located in different places are needed. Therefore in their system two measurements are performed. The cases discussed so far can regarded as 2D cases. In Akbarally & Kleeman, (1995) the described method was extended to 3D case. The constructed sonar system consisted of five ultrasonic transducers. A binaural sonar system for object differentiation was presented in Ayrulu et al., (1997). It consisted of two receivers and two transmitters. The sonar system was able to measure TOF and an echo amplitude. Objects features were generated as being evidentially tied to degrees of belief which were subsequently fused by employing multiple logical sonars at different geographical sites. Feature data from multiple logical sensors were fused with Dempster-Shafer rule of combination to improve the performance of classification by reducing perception uncertainty. Dempster-Shafer fusion results were contrasted with the results of combination of sensor beliefs through simple majority vote. A different approach is presented in Heale & Kleeman, (2001). It is based on the Maximum Likelihood Estimation technique. To perform the localization and classification task a real time DSP-based sensing module was constructed. It made possible to apply a double pulse coding method. This approach was extended in order to take into account robot movement Kleeman, (2004).

The object position determination in 3D coordinate system is a bit more complicated. It can be shown that to distinguish edge, corner, wall and point-like object, measurements performed by using at least three transmitters and receivers are needed Kreczmer, (2006). If they can also work as receivers then the system can be restricted up to three ultrasonic transducers. It seems to be the minimal configuration. This kind of system was also applied by Li & Kleeman, (1995) to differentiate walls and corners. In Jimenez et al., (2005) a classification method based on

the principal component-analysis technique was proposed. A sonar system which was used in implementation of the method consisted of eight ultrasonic transducers. In Ochoa et al., (2009) approach was improved and the sonar system reduced two four transducers.
Another crucial point in the problem of object localization and differentiation is the accuracy of echo detection. The currently reported precision is about 1*mm* e.g. Egaa et al., (2008) Angrisani & Moriello, (2006) which is satisfying for many applications.

3. Object localization by binaural sonar system

The basic technique for object localization using TOF is the well known triangulation method. The system applying this method has to consist of at least two receivers and a single emitter. It can be also used a single receiver and a transducer which can work as an emitter and a receiver. To reduce the error of object position determination both receivers have to be placed as far as possible from each other. But there are additional conditions which limit the distance between them. If the distance is too big, it can happen that the echo will be received mostly by only a single receiver. Another case arises when there are a lot of objects in the robot environment. To large baseline of the sonar system can cause that receivers register an echo which hasn't been produced by the same object. This is the reason while the distance between sonars cannot be uniquely determined. It must be adjusted to an environment in which the robot operates, and to the expected maximal range of distance to an object which should be localized. The length of the baseline must be also adjusted to measurement errors.
The simples case of object position determination is an edge or a narrow pole (see fig. 3). The

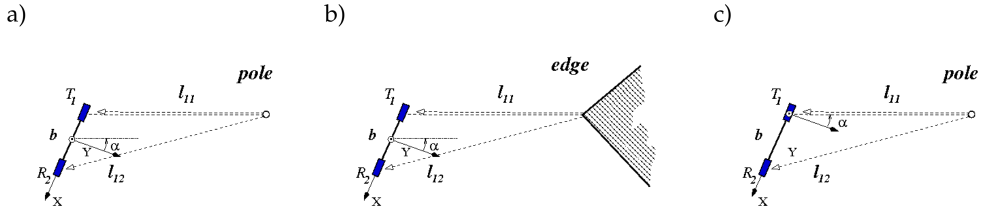


Fig. 3. The bird's-eye view of the signal paths for a) a pole, b) an edge, c) a pole and the coordinate system placed at T_1

distance of a path flown by an ultrasonic wave from the emitter T_1 and back to T_1 switched to the receiver mode, is l_{11}. The length of the signal path from T_1 to the receiver R_2 is l_{12}. Applying a triangulation method the Cartesian coordinates of the object can be determined using simple formulae

$$x = \frac{1}{2b} l_{12}(l_{11} - l_{12}), \qquad y = \frac{1}{2}\sqrt{l_{11}^2 - \frac{1}{b^2}(l_{12}(l_{11} - l_{12}) + b^2)^2}. \tag{1}$$

They are derived for the local coordinate system placed in the middle of the sonar system (see fig. 3a,b). The polar coordinates of the object are determined by formulae

$$r = \frac{1}{2}\sqrt{(l_{11} - l_{12})^2 + l_{12}^2 - b^2}, \qquad \alpha = \arcsin \frac{l_{12}(l_{11} - l_{12})}{b\sqrt{(l_{11} - l_{12})^2 + l_{12}^2 - b^2}}. \tag{2}$$

It is assumed that the angle α is measured from the axis OY in the anticlockwise direction. In the coordinate system of the transmitter (see fig. 3c) the formula for r becomes extremely simple. All formulae in this coordinate system are as follows

$$x = \frac{1}{2b}(l_{12}(l_{11} - l_{12}) + b^2), \qquad y = \frac{1}{2}\sqrt{l_{11}^2 - \frac{1}{b^2}(l_{12}(l_{11} - l_{12}) + b^2)^2}.$$

$$r = \frac{l_{11}}{2}, \qquad \alpha = \arcsin\frac{1}{bl_{11}}(l_{12}(l_{11} - l_{12}) + b^2). \tag{3}$$

For a wall the paths of an received echos are a bit more complicated (see fig. 4a). The inclination angle of the signal coming back to T_1 is always equal to $90°$. For the signal received by R_2 the inclination and reflection angle are the same. This is due to the assumption that irregularities of an object surface are much smaller than the wave length of the emitted signal. The analyze is much more easier if the approach of the virtual image of the sonar system is

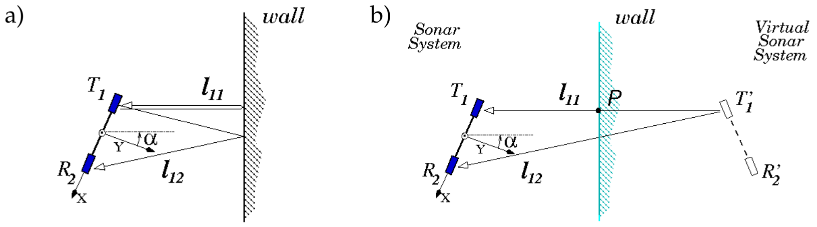


Fig. 4. a) The bird's-eye view of the signal paths for a wall. b) The symmetrical image of the sonar system allows to simplify the signal paths analysis

applied. The construct of virtual images is borrowed from an optical context and is used by many researches Peremans et al., (1993)Kleeman & Kuc, (1995)Heale & Kleeman, (2001). The virtual image of a transducer in a plane is obtained by reflecting the true position of the transducer about the plane. In this way the wall introduce a plane symmetry (see fig. 4b). The location of the point P can be easily determined. Its Cartesian coordinates in the local coordinate system of the binaural sonars range-finder are

$$x = \frac{1}{4b}(l_{11}^2 - l_{12}^2 - b^2), \qquad y = \frac{1}{2}\sqrt{l_{11}^2 - \frac{1}{4b^2}(l_{11}^2 - l_{12}^2 + b^2)^2}. \tag{4}$$

In the polar coordinate system the components of coordinates can be expressed as follows

$$r = \frac{l_{12}}{2}, \qquad \alpha = \arcsin\frac{1}{2bl_{12}}(l_{12}^2 - l_{11}^2 + b^2). \tag{5}$$

This time the formula for r is simple in the both coordinate systems i.e. the coordinate system placed in the middle of the baseline (see above) and the coordinate system combined with the sonar T_1. The formulae expressed in the last mentioned coordinate system are presented below

$$x = \frac{1}{4b}(l_{11}^2 - l_{12}^2 + b^2), \qquad y = \frac{1}{2}\sqrt{l_{11}^2 - \frac{1}{4b^2}(l_{11}^2 - l_{12}^2 + b^2)^2},$$

$$r = \frac{l_{11}}{2}, \qquad \alpha = \arcsin\frac{1}{2bl_{11}}(l_{12}^2 - l_{11}^2 - b^2).$$

Because l_{11} is perpendicular to a surface of a wall the straight line equation of a vertical cast of a wall can be determined.
The next very characteristic reflector is a corner. It is an example of surfaces arrangement which is the source of double reflections (see fig. 5a). The virtual image of a transducer in a

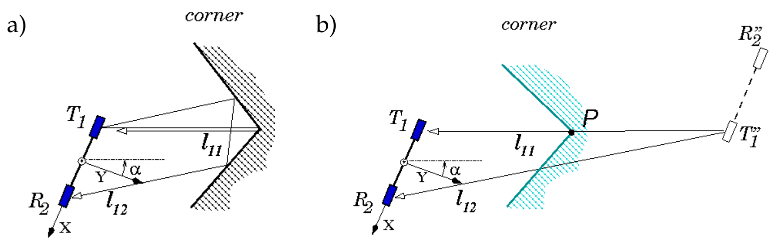


Fig. 5. a) The bird's-eye view of the signal paths for a corner. b) Signal path after creating a virtual sonar system being an image of the real system using axial symmetry

corner is obtained by reflecting about one plane and then the other which results in a reflection about the line of intersection of the planes. Thus finally the axial symmetry is obtained. This kind of symmetry is also obtained for the edge case. The only difference is that drawing signal path from a virtual sonar to a real one the path must always cross the edge (see fig. 7a). In this sens it isn't the same technique which is used for a wall or a corner. Considering 2D case the coordination of the point P can be determined (see fig. 5b). The obtained measurements results don't allow to determine the location of both walls. It doesn't depend on the number of senders and receivers. It is clear when two different corner orientations are considered (see fig. 6). For both orientations the same virtual image of sonar system is obtained. The position

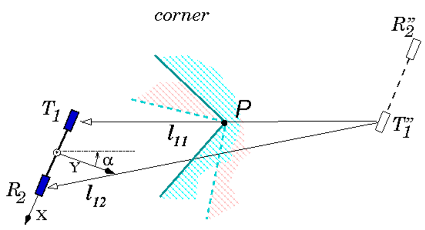


Fig. 6. The same virtual image of the sonar system is created for two different orientation of the corner

of the point P can be computed using (4) and (5). It means that the formulae can be applied which are used for a wall.

4. Object classification

Because the formulae allowing to determine object location are different for edges, walls and corners, to determine correctly the object position first it should be recognize and properly classified. Data obtained from a single measurement aren't enough to distinguish objects discussed in this section. It was shown that at least two measurements are necessary by using emitters located at different places Kleeman & Kuc, (1995). It can be done using the binaural sonar system. To do so it is necessary to replace the receiver R_2 with a transducer T_2 working as an emitter and a receiver. In this form the sonar system makes possible to perform two measurements. The first is done by using T_1 as a transmitter and the second is executed by using T_2. During both measurements the transducers are switched into receiving mode after the signal emissions. Signal paths for all cases are shown in fig. 7. In all sketches of signal

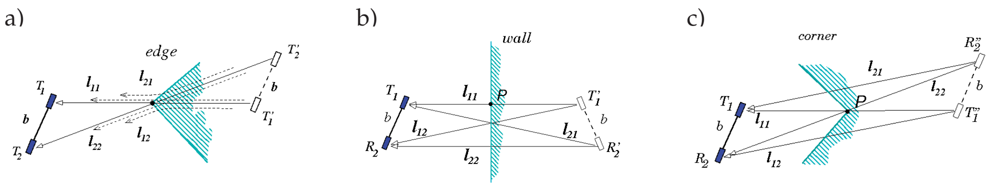


Fig. 7. The virtual images of the sonar system and signal paths for all cases. The edge case differs form the cases of a wall and a corner because signal path from a virtual sonar must cross the point marked the edge

paths the technique of virtual images is exploited. The edge case doesn't follow exactly this technique. But this drawing makes easier to notice some relations.
Considering geometrical figures created by signal paths and sonar systems and their virtual images for each case a single condition can be found. Their are as follows

$$\begin{array}{lcrcl} \textit{edge} & \rightarrow & l_{12}+l_{21}-l_{11}-l_{22} & = & 0, \\ \textit{wall} & \rightarrow & (l_{12}+l_{21})^2-4b^2-4l_{11}l_{22} & = & 0, \\ \textit{corner} & \rightarrow & (l_{12}+l_{21})^2+4b^2-2l_{11}^2-2l_{22}^2 & = & 0. \end{array} \tag{6}$$

It can be added another condition $l_{12}-l_{21}=0$. But it isn't combined directly with arrangement of signal paths. It is rather a general feature. The presented condition can be used for object distinguishing. In Heale & Kleeman, (2001) the Maximum Likelihood Estimation classification approach has been applied which was based on the conditions presented above. But if an measurement error can be limited to a certain range a simpler method can be proposed. At the beginning let us consider a possibility of using of the edge condition as a criterion of reflector classification, i.e.

$$C_e(O_i,b)=l_{11}+l_{22}-l_{12}-l_{21}.$$

where O_i is an object being source of echos, b is the distance between sonars. l_{11},l_{22}, l_{12}, l_{21} are measured distances which are the functions of object type, its location and the length of the baseline b.
Analyzing fig. 7a and fig. 7c it can be easily noticed that for a corner it must be $C_e(O_c,b)\geq 0$. But the condition $C_e(O_c,b)=0$ is met only for the orientation of the sonar system: $\alpha=-90°$ or $\alpha=90°$. These configurations aren't needed to be taken into account. It is caused by the fact that if a direction to an object is far from the direction of the transmitter acoustic axis then an amplitude of an emitted signal is equal to 0 or almost 0. Thus for all reasonable configurations the condition $C_e(O_c,b)>0$ must be held.
For the wall case fig. 7a and fig. 7b should be considered. It can be found out that $C_e(O_w,b)\leq 0$. But $C_e(O_w,b)=0$ is in the same situations like for a corner. For the same reasons it can be assumed that for all possible configurations it should be $C_e(O_w,b)<0$. The deduced features of $C_e(O_i,b)$ allows it to be a good candidate for the criterion distinguishing edges, walls and corner.

To prove the discussed feature the function $C_e(O_w, b)$ can be derived using the parameterization by d and α where d is the distance to a wall from the sonar system and α is its orientation in relation to the wall (see fig. 3)

$$C_e(O_W, b) = C_{e,W}(d, \alpha, b) = 2\sqrt{4d^2 + b^2 \cos^2 \alpha} - 4d. \tag{7}$$

The same type of parameterization can be used for a corner. It gives

$$C_e(O_C, b) = C_{e,C}(d, \alpha, b) = 4d - \sqrt{(2d - b\sin\alpha)^2 + b^2\cos^2\alpha} - \sqrt{(2d + b\sin\alpha)^2 + b^2\cos^2\alpha}. \tag{8}$$

The example of the function plots for $C_e(O_{C_k}, b)$ and $C_e(O_{W_k}, b)$ is presented in fig. 8. These

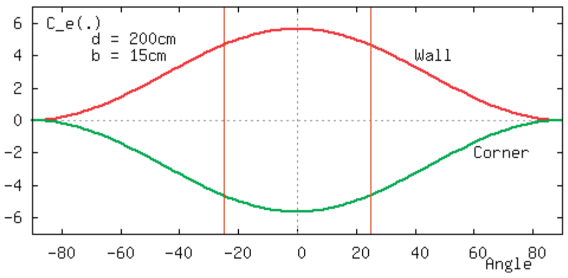


Fig. 8. The values of the criterion C_e for a wall and a corner. The charts are drawn for a system placed $2m$ from an object and sonars placed in the distance $15cm$ to each other

plots confirm the previous deduction based on geometric shapes of the paths. The chart shows that for $|\ \alpha\ | > 60°$ the values of $C_e(O_{C_k}, b)$ and $C_e(O_{W_k}, b)$ are very close to 0. But the full range of orientation isn't necessary to take into account. Because the width of the emitted beam is restricted to about $40° \sim 50°$, the range of considered orientations can be confined to $[-25°, 25°]$. It gives an additional advantage because in this range the functions reach values which are very close to the extrema ones. That makes possible to distinguish all considered objects.

It can be expected that it is easier to distinguish the object when it is close than it is far. In other words it means that the values of $C_e(O_E, b)$, $C_e(O_W, b)$ and $C_e(O_W, b)$ should be close to each other when the distance to objects is small. The difference should be increased when the distance is large. Because $C_e(O_E, b)$ for all distance equals to 0, the presented feature means that for large distances, values of $C_e(O_W, b)$ and $C_e(O_C, b)$ are more apart from 0. This feature can be observed on charts presented in fig. 9. The analogous feature can be noticed while the distance b between sonars is considered. The larger baseline b the more extreme values by $C_e(O_W, b)$ and $C_e(O_C, b)$ are reached (see fig. 10).

In the following part the stable conditions in the environment are assumed. It means that in the surrounding where the sonar system operates there are no wind, no very hot spots etc. It allows us to assume that the maximal measurement error Δl can be estimated and is the same for all l_{ij}. Thus the error of $C_e(O_i, b)$ is

$$\Delta C_e(O_i, b) = \Big(\Big|\frac{\partial C_e}{\partial l_{11}}\Big| + \Big|\frac{\partial C_e}{\partial l_{12}}\Big| + \Big|\frac{\partial C_e}{\partial l_{21}}\Big| + \Big|\frac{\partial C_e}{\partial l_{22}}\Big|\Big)\Delta l + \Big|\frac{\partial C_e}{\partial b}\Big|\Delta b = 4\Delta l.$$

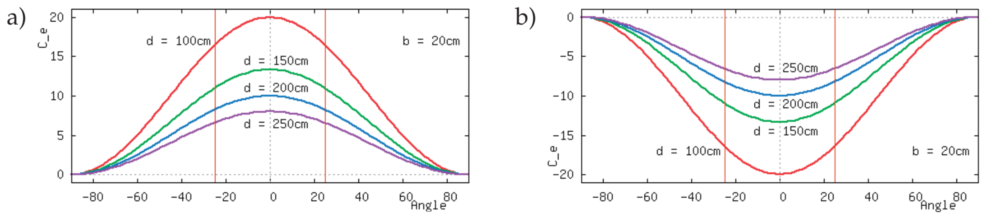


Fig. 9. The values of the criterion C_e for a wall and a corner. The charts are drawn for a system placed at different distances from an object and sonars separation distance $b = 20cm$

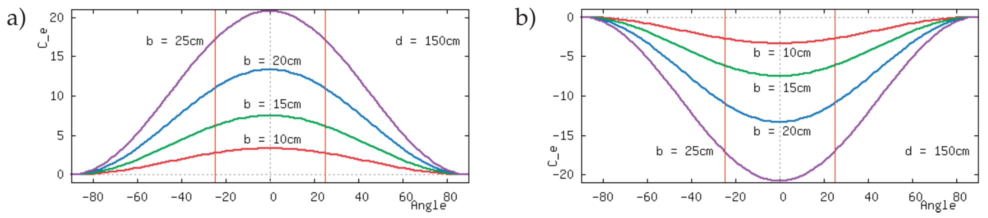


Fig. 10. The values of the criterion C_e for a wall and a corner. The charts are drawn for a system placed at 1.5m from and object and different distances between sonars

It means that if an object must be properly distinguished at the distance d_{max}, the value b must be enough big in order to obtain

$$C_{e,W}(d_{max}, \alpha_{max}, b) > 8\Delta l \quad \wedge \quad C_{e,C}(d_{max}, \alpha_{max}, b) < 8\Delta l.$$

where α_{max} is the biggest value of the assumed orientation range. In the considered example it has been assumed $\alpha_{max} = 25°$. The equations presented above is the necessary condition but not sufficient one. The example of proper choice of b for $d_{max} = 1.5m$ is presented in fig. 11. It shows the drawing of $C_{e,W}$ and $C_{e,C}$ with *tunnels* of acceptable values. The classification

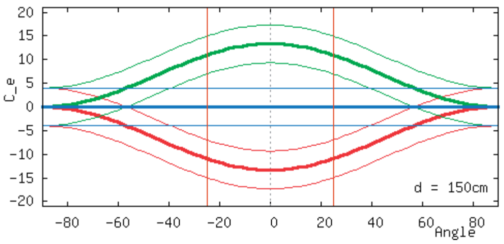


Fig. 11. The values of the criterion C_e for a corner, an edge and a corner. In the chart, ranges of acceptable values in the sens of error measurement are marked

procedure which includes sufficient conditions can be described by the rules:

$$\begin{array}{llll} if \quad C_e(O_i, b) & \in & (-4\Delta l,\ 4\Delta l) & \Rightarrow \text{edge}, \\ if \quad C_e(O_i, b) & \in & (C_{e,W}(d,\alpha,b) - 4\Delta l,\ C_{e,W}(d,\alpha,b) + 4\Delta l) & \Rightarrow \text{wall}, \\ if \quad C_e(O_i, b) & \in & (C_{e,C}(d,\alpha,b) - 4\Delta l,\ C_{e,C}(d,\alpha,b) + 4\Delta l) & \Rightarrow \text{corner}, \\ & & \quad otherwise & \Rightarrow \text{unknown}. \end{array} \tag{9}$$

This set of rules seems to contain a dead loop. To classify an object properly its position must be known. But to determine properly the position of the object it must be first classified. Fortunately the difference of paths length between the edge and wall case is small. Therefore the procedure applied for an edge can be also used for a wall and a corner. The error of the angle α for wall localization is about $0.5°$ while the distance between sonars is $8cm$ and the wall is at the distance $2m$. The error is increased to about $1°$ when the distance between sonars is increased up to $16cm$. The error of distance computing is negligible. In this way the very good approximation of the real position can be obtained. But it is still possible to improve it. It can be noticed that for a wall when T_1 is used as an emitter, the hypothetical reflecting edge is a bit on the left in relation to the correct direction to the wall (see E_1 in fig. 12a). When T_2 is used as an emitter, the the hypothetical reflecting edge is on the opposite site almost with the same angle (see E_2 in fig. 12a). For a corner the final result is similar. The only difference is that

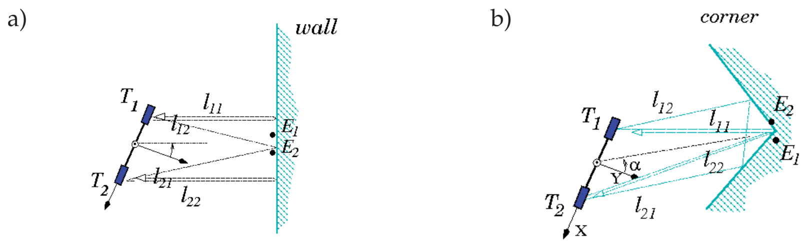


Fig. 12. The determined positions of hypothetical objects when the triangulation procedure is applied for measurements obtained for the wall case

the positions of hypothetical edges are reversed comparing with the wall case. This feature is exploited in Heale & Kleeman, (2001) for object distinguishing. The important disadvantage of the method is that it doesn't allow easily to take into account errors of measurements and set a proper margin of acceptable values.

The computed locations of hypothetical edges make possible quite precisely to determine the position of the object. Because the previously discussed errors of the hypothetical edges have opposite value, it is enough to compute the mean of these locations to obtain the position of the object. It reduces the error of the angle α to about $0.1°$. This is for the worst case when object is located at the distance $2m$ with the bearing angle $25°$. The best case is for bearing $0°$. It gives exactly symmetrical configuration which cancels the error. Because even for the worst case the bearing error is very small, it can be neglected.

This procedure also reduces the error of bearing determination caused by measurements error of l_{ij}. If the interval between the successive sonars firing T_1 and T_2 is short, then distance measurement is highly correlated Heale & Kleeman, (2001). Therefore the measurement error is usually much lower than the assumed error of a single separate measurement. Nevertheless this kind of the bearing error must be taken into account for a wall and a corner classification because $C_{e,W}$ and $C_{e,C}$ depend on α. In this sense it isn't needed to be considered for $C_e(O_E, b)$. The criterion value for an edge is constant for each orientation of the sonar system. Thus it doesn't depend on the bearing angle.

The error of the α determination can cause that some acceptable values can be shifted outside the *tunnel* of the expected values. Moreover, some unacceptable values can be moved into the tolerance *tunnel* of measured values. Thus if we want to be sure that the classification is proper in the sens of assumed measurements error, the *tunnels* of acceptable values must

be narrowed. Considering this part of acceptable range which is shifted outside the tunnel, it is also necessary to widen the tunnel. The area between the border of the narrowed and widened tunnel contains values for which an object can be a wall but not for sure (in the sens of the assumed measurement error). To clinch it a next measurement has to be perform. The area outside the tunnel contains values which indicate that an object isn't a wall for sure.
Taking into account the aforementioned arguments it allows us to create the border of the narrowed tunnel. The top and bottom border of this area can be defined as follows

$$\overline{C}_{W,nar}(d,\alpha,b) = \begin{cases} C_{e,W}(d,\alpha-\Delta\alpha,b)+4\Delta l & \alpha<\Delta\alpha \\ C_{e,W}(d,0,b)+4\Delta l & \alpha\in[-\Delta\alpha,\Delta\alpha] \\ C_{e,W}(d,\alpha+\Delta\alpha,b)+4\Delta l & \alpha>\Delta\alpha \end{cases}$$

$$\underline{C}_{W,nar}(d,\alpha,b) = \begin{cases} C_{e,W}(d,\alpha+\Delta\alpha,b)-4\Delta l & \alpha<\Delta\alpha \\ C_{e,W}(d,0,b)-4\Delta l & \alpha\in[-\Delta\alpha,\Delta\alpha] \\ C_{e,W}(d,\alpha-\Delta\alpha,b)-4\Delta l & \alpha>\Delta\alpha \end{cases}$$

In similar way the border of the widened tunnel can constructed. The same procedure and analogically definitions can be used to construct the tunnels of acceptable values for a corner. This approach suffers the main important disadvantage. The value of $\Delta\alpha$ also depends on the measured distances l_{ij}. Furthermore the formula of $\Delta\alpha$ is very complicated. Thus such an approach cannot be easily implemented.
The advantage of using C_e is that for an edge it doesn't depend on α. Therefore their values doesn't depend on $\Delta\alpha$. Unfortunately, as it has been shown, the same arguments cannot be used for corner and wall cases.
In the same way as the edge criterion has been defined, the criterion for a wall can be constructed. Using the second equation from (6) the criterion can be defined as follows

$$C_w(O_i,b) = (l_{12}+l_{21})^2 - 4b^2 - 4l_{11}l_{22}.$$

The analogical criterion can be defined for a corner using the third equation from (6)

$$C_c(O_i,b) = (l_{12}+l_{21})^2 + 4b^2 - 2l_{11}^2 - 2l_{22}^2.$$

The error of C_w can be approximated by the formula

$$\Delta C_w = \Big(\Big|\frac{\partial C_w}{\partial l_{11}}\Big| + \Big|\frac{\partial C_w}{\partial l_{12}}\Big| + \Big|\frac{\partial C_w}{\partial l_{21}}\Big| + \Big|\frac{\partial C_w}{\partial l_{22}}\Big|\Big)\Delta l + \Big|\frac{\partial C_w}{\partial b}\Big|\Delta b = 4(l_{11}+l_{12}+l_{21}+l_{22})\Delta l + 8b\Delta b. \tag{10}$$

Expressing the measured lengths of signal paths l_{ij} as functions of d and α for the wall case, the following formulae are obtained

$$\begin{aligned} l_{11} &= 2d + 2b\sin\alpha, \\ l_{22} &= 2d - 2b\sin\alpha, \\ l_{12} = l_{21} &= \sqrt{4d^2 + b^2\cos^2\alpha}. \end{aligned}$$

Applying these expressions to the formula (10) it gives

$$\Delta C_w = (16d + 8\sqrt{4d^2 + b^2\cos^2\alpha})\Delta d + 8b\Delta b.$$

Because $d \gg b$ and $\Delta d \geq \Delta b$ then

$$\Delta C_w \simeq 32d\Delta l.$$

Thus ΔC_w can be considered as independent on α. It can be noticed that C_w has the similar feature for a wall as C_e for an edge (see fig. 13a). Considering C_c the same formula is obtained

$$\Delta C_c = \left(\left|\frac{\partial C_c}{\partial l_{11}}\right| + \left|\frac{\partial C_c}{\partial l_{12}}\right| + \left|\frac{\partial C_c}{\partial l_{21}}\right| + \left|\frac{\partial C_c}{\partial l_{22}}\right|\right)\Delta l + \left|\frac{\partial C_c}{\partial b}\right|\Delta b = 4(l_{11} + l_{12} + l_{21} + l_{22})\Delta l + 8b\Delta b.$$

This time the measured lengths of signal paths l_{ij} have to be expressed for the corner case as functions of d and α. It gives formulae

$$\begin{aligned} l_{11} = & \sqrt{4d^2 + b^2 \sin\alpha}, \\ l_{22} = & \sqrt{4d^2 - b^2 \sin\alpha}, \\ l_{12} = l_{21} = & \ 2d. \end{aligned} \tag{11}$$

Direct substitution to (11) and computing derivations yields

$$\Delta C_c = (16d + 4\sqrt{4d^2 + b^2 \sin\alpha} + 4\sqrt{4d^2 - b^2 \sin\alpha})\Delta l + 8b\Delta b.$$

Using the same arguments as before i.e. $d \gg b$ and $\Delta d \geq \Delta b$, the final approximation is obtained

$$\Delta C_c \simeq 32d\Delta l.$$

It allows us to draw charts of C_w and C_c and mark *tunnels* of acceptable values. It is done in the same way as it has been done for C_e. Fig. 13 presents these charts. It can be noticed

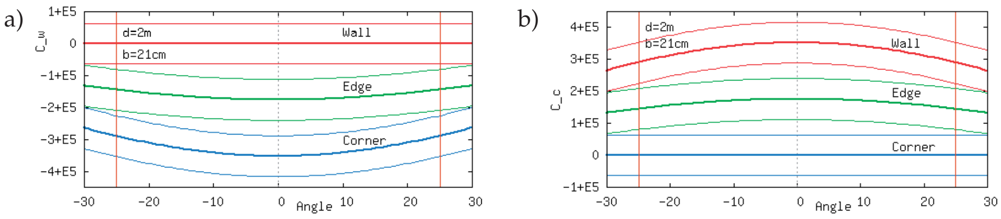


Fig. 13. Charts of C_w and C_c being drawn for a wall, an edge and a corner. Around them ranges of acceptable values for proper classification are marked

that the range of the proper separation of all cases is preserved (compare fig. 13 and fig. 14). Therefore it is enough to analyze only a single criterion in order to determine the initial length of the baseline b which guarantees the correct classification of the considered objects at a given

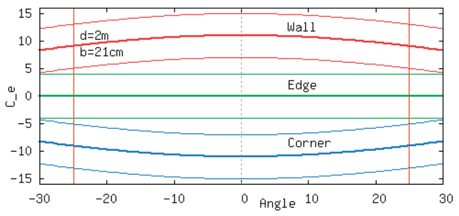


Fig. 14. Charts of C_e being drawn for a wall, an edge and a corner. Around them ranges of acceptable values for proper classification are marked

distance. But to classify an object all criterion have to be taken into account in order to avoid any ambiguity. The procedure of classification can be written as a set of rules

$$\begin{array}{lllll} \mid C_e(O_i) \mid \leq 4\Delta l \ \wedge & \mid C_w(O_i) \mid > 32d\Delta l \ \wedge & \mid C_c(O_i) \mid > 32d\Delta l & \Rightarrow & \textit{edge}, \\ \mid C_e(O_i) \mid > 4\Delta l \ \wedge & \mid C_w(O_i) \mid \leq 32d\Delta l \ \wedge & \mid C_c(O_i) \mid > 32d\Delta l & \Rightarrow & \textit{wall}, \\ \mid C_e(O_i) \mid > 4\Delta l \ \wedge & \mid C_w(O_i) \mid > 32d\Delta l \ \wedge & \mid C_c(O_i) \mid \leq 32d\Delta l & \Rightarrow & \textit{corner}, \\ & & \text{otherwise} & \Rightarrow & \text{unknown}. \end{array} \tag{12}$$

It is worth to note that the procedure doesn't require to use trigonometric functions or square root operations. Therefore it can be implemented using a low cost controller which doesn't support float point arithmetic.
Another important feature is combined with the set of equations (6). It can be noticed that all of them it is possible to obtain from a single general equation

$$(l_{12} + l_{21})^2 - (l_{11} + l_{22})^2 + \rho\Big((l_{11} - l_{22})^2 - 4b^2\Big) = 0$$

where $\rho \in [-1, 1]$. The equation for a corner is obtained for $\rho = -1$. The values 0 and 1 make possible to obtain the equations for an edge and a wall respectively. Thus instead of using the rules (12) the ρ value can be computed and in this way an object can be classified.

5. Simplified triangulation method for object location

The method of object distinguishing presented in the previous section has very important advantage. It can be implemented without using float point arithmetic. This method doesn't use trigonometric functions or square root operations. In this way low cost processors can be used to create a more intelligent sensor. But this advantage is lost when we want to combine this approach with the location method based on triangulation procedure. The formulae (1) and (2) requires such a function and an operation. In this section a method is presented which makes possible to omit this problem. It allows to reduce the necessary computation power and in this way it makes possible to construct a cheap intelligent sensor which can determine the location of an object and can classify it.
The presented approach is based on the method described in Kuc, . The main idea of the approach consists in determining a pass distance between an obstacle and a mobile robot moving along a straight line. Successive results of measurement are used in order to compute an object position. The pass distance y_p (see fig. 15a) can determined using the equation (13).

$$y_p \quad = \quad \sqrt{r_d^2 - x_p^2} \tag{13}$$

Assuming that the measurements are performed in placed which are regularly spread along the straight line (see fig. 15b) the pass distance can expressed by the formula (14). Building this formula it is also assumed that the object is passed by at the point determined by $k = 0$.

$$y_{p,k} \quad = \quad \sqrt{r_k^2 - (kd_s)^2} \tag{14}$$

Considering a sequence of three places the computed values of y_p can be expressed by the set of equations (15).

$$\begin{array}{rcl} y_{p,k} & = & \sqrt{r_k^2 - (kd_s)^2} \\ y_{p,k-1} & = & \sqrt{r_{k-1}^2 - ((k-1)d_s)^2} \\ y_{p,k-2} & = & \sqrt{r_{k-2}^2 - ((k-2)d_s)^2} \end{array} \tag{15}$$

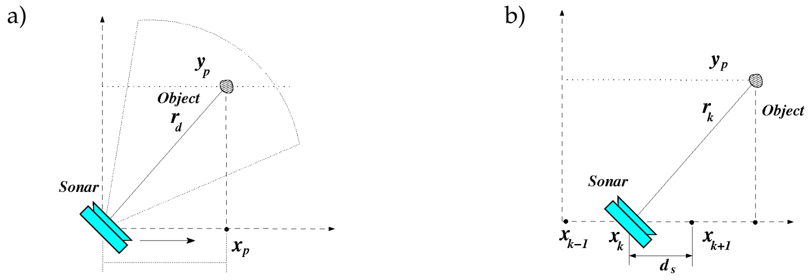


Fig. 15. a) A robot equipped with an ultrasonic sonar moves along a straight line and passes by an object. b) The measurements of the distance to an object are performed at the points regular spread along the robot path

It is more convenient to compute y_p^2 instead of y_p. The differences of y_p^2 for each two following points are presented by formulae (16)

$$\begin{aligned} y_{p,k}^2 - y_{p,k-1}^2 &= r_k^2 - r_{k-1}^2 - 2kd_s^2 + d_s^2 \\ y_{p,k-1}^2 - y_{p,k-2}^2 &= r_{k-1}^2 - r_{k-2}^2 - 2kd_s^2 + 3d_s^2 \end{aligned} \tag{16}$$

The computed values $y_{p,k}, y_{p,k-1}, y_{p,k-2}$ should meet the condition $y_{p,k} = y_{p,k-1} = y_{p,k-2} = y_p$. In this way we obtained.

$$\begin{aligned} 0 &= r_k^2 - r_{k-1}^2 - 2kd_s^2 + d_s^2 \\ 0 &= r_{k-1}^2 - r_{k-2}^2 - 2kd_s^2 + 3d_s^2 \end{aligned} \tag{17}$$

These formulae allow us to determine k parameter.

$$\begin{aligned} k &= \tfrac{1}{2d_s^2}(r_k^2 - r_{k-1}^2) + \tfrac{1}{2} \\ k &= \tfrac{1}{2d_s^2}(r_{k-1}^2 - r_{k-2}^2) + \tfrac{1}{2} + 1 \end{aligned} \tag{18}$$

The general form of a formula exploiting r_{k-l} and r_{k-l-1} is presented below

$$k = \frac{1}{2d_s^2}(r_{k-l}^2 - r_{k-l-1}^2) + \frac{1}{2} + l. \tag{19}$$

This formula can be still simplified while we take into account a method of distance determination. The most popular method is based on time measurement of a signal flight (TOF). Because the time is measured by digital clock, the measured distance can expressed as follows

$$r = \frac{c_s \Delta t}{2} = \frac{c_s(p\Delta\tau)}{2} = p\frac{c_s\Delta\tau}{2}$$

where $\Delta\tau$ is an elementary slice of time measured by clock, m is the number of elementary time slices. We can arbitrary choose the duration of the elementary slice $\Delta\tau$. Thus we can choose the value of the slice in order to meet the equation (20)

$$d_s = \frac{c_s\Delta\tau}{2}. \tag{20}$$

It allows us to simplify the formula (19) as follows

$$k = \frac{1}{2}(p_{k-l}^2 - p_{k-l-1}^2 + 1) + l \tag{21}$$

where m_{k-l} and m_{k-l-1} are the numbers of time slices counted while the distances r_{k-l} and r_{k-l-1} are measured. It is worth to note that all elements of the equation (21) are integer numbers. In this way the software implementation can be simplified and it makes even possible a hardware implementation of the approach.
This kind of discretization introduce an error of distance measurement equal: $\Delta r = d_s$. When d_s is big (e.g. $10cm$ or more) this cannot be accepted. It can be reduced by choosing the smaller time quantum $\Delta\tau$ as follows

$$d_s = n\frac{c_s \Delta\tau}{2}.$$

Applying it to (19) it gives

$$k = \frac{1}{2n^2}(q_{k-l}^2 - q_{k-l-1}^2 + 1) + l.$$

The advantage of this formula is obtained when n is a power of 2. It allows to reduce the division arithmetic operation to register shifting.

5.1 Multi-sonar system

The important advantage of the method presented in the previous section is the elimination of artifacts which are sensible to a change of a sender and detector position. Using the method a sonar must be moved within the same distance step by step. When a sonar is mounted on a mobile robot it can be obtained during robot motion. But this solution introduce additional source of errors i.e. the distance measurement of a robot movement. It can be avoid when a set of sonars is used, instead a single moving sonar (see fig. 16). The sonars can be mounted within a regular shift and the error can be drastically reduced. The sonars arrangement cre-

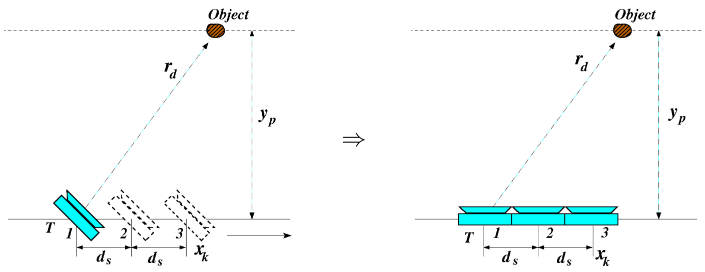


Fig. 16. Instead of regular sonar displacement a sonar array can be used

ates a sonar array. In the simples form it is possible to reduce it to a tri-aular sonar system. Considering the presented method the system can be treated as a double binaural system. The second pair of sonars is used in order to verify the results obtain by the first pair. It shows that the method presented in Kuc, (2007) can be transformed to the known approach which was applied to reduce false sonars reading Kreczmer, (1998). However, the method described

in Kuc, (2007) offers new important benefits in the sense of discretization presented in the previous section. It allows to speed up calculations and makes results more robust.
The method proposed in the previous section introduces an additional error to calculations of the distance y_p. It is caused by discretization. It should be noticed that the step of discretization d_s is quit big. Comparing the approach based on moving a single sonar and tri-aular system it can be shown that the last one has an important advantage. It is due to the orientation of discretized coordinate axis to an object and a sonar acoustic axis. The fig. 16 shows the case when the situation for both systems are exactly equivalent. Because the width of a sonar beam isn't very wide (30° for Polaroid transducers) in order to be noticed an object cannot be far from the acoustic axis of a sonar. When a sonar is pointed towards the movement direction (see fig. 17a), an object cannot be very far from this direction. Considering a tri-aural sonar system or in general a linear array system an object cannot be far from the acoustic axis of the system (see fig. 17b).

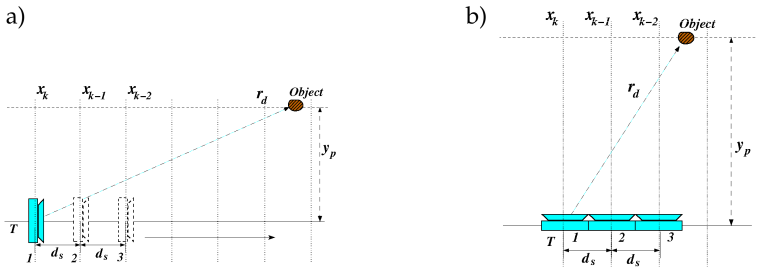


Fig. 17. a) When a sonar is pointed into direction of its movement, an object cannot be far from this direction. Discretization of the distance x_p is applied along the movement direction. b) The object cannot be far from the acoustic axis of the sonar linear array. Discretization of the distance x_p is along the sonar array

The main difference of these two cases is that for a moving sonar $y_p < x_p$ or even $y_p \ll x_p$. The opposite relation is for a multi-sonar system i.e. $y_p \gg x_p$. This feature has very strong impact on an error value of y_p.
Considering the case of moving sonar pointed into the direction of the movement it can be noticed that an error of k being equal to 1 can cause a very big change of the value y_p (see fig. 18a). The same situation considered for a multi-sonar system causes much smaller change of y_p (see fig. 18b) and in consequence much lower error value.
The type of discretization presented in fig. 18b gives an additional advantage. When a value of y_p is needed to be established in order to locate an object in a robot surrounding, the formula (14) should be used. But the value of r_k can be also digitalized and in this way the formula (14) isn't needed any longer to show where the object is. Digitizing x_p and r_k a kind of grid is created (see fig. 19b) which is a mix of Cartesian and polar coordinates discretization. Creating the same type of grid for a sonar pointed towards its movement direction, much bigger cells are obtained. Moreover the angle coordinate is much more fuzzy (see fig. 19a) than for the case of discretization applied to a multi-sonar system. The presented approach to processing of measurement data obtained from a multi-sonar system makes possible to reduce necessary resources of the system controller. Using the final discretized representation of data and a local environment the controller can perform all calculations using only integer data. Which is the very important benefit of the described method.

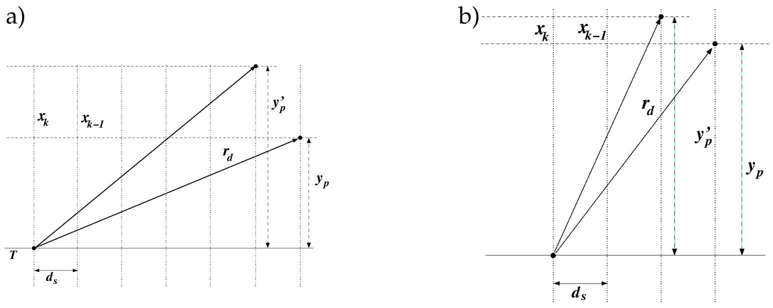


Fig. 18. a) While $y_p \ll x_p$, a small change of k can cause a very big change of value y_p. b) While $y_p \gg x_p$, the same change of k can cause a very little change of value y_p

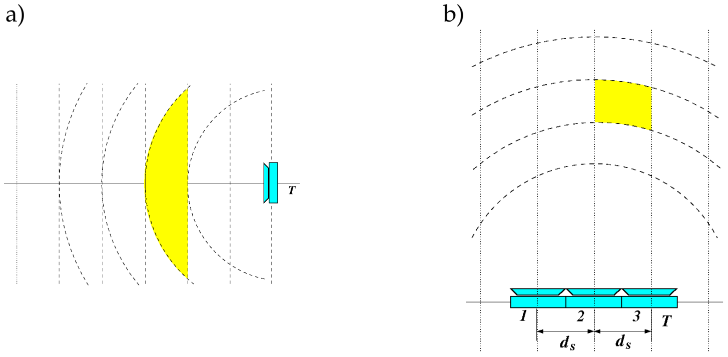


Fig. 19. The mixed Cartesian and polar coordinates discretization for a) a sonar pointed towards its movement direction, b) a multi sonar system

Considering the binaural system it should be assumed that $b = d_s$. But it creates very wide grid cells which is unacceptable (see fig. 20). Fortunately it can be easily solved. Because

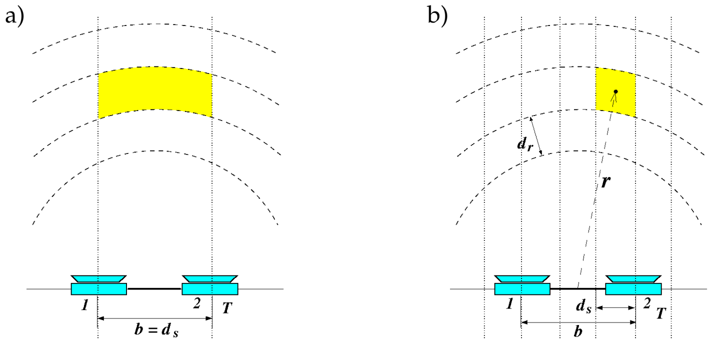


Fig. 20. a) The grid with the big discretization step being equaled to the length b of the baseline of the binaural sonar system. b) More fine discretization where b is a multiplication of d_s.

creating more fine discretization it is obtained

$$b = md_s \quad \Rightarrow \quad \tilde{k}_s = \frac{k}{m}.$$

where m is an positive integer number. It determines the precision of discretization (see fig. 20). Finally it gives

$$\tilde{k}_s = \frac{1}{2n^2m}(q^2_{k-l} - q^2_{k-l-1} + m). \tag{22}$$

In this formula comparing it with the formula (19) the parameter l is set to 0 because $\tilde{k}$ is assumed to 0 at the position of the left sonar. Using the binaural sonar system it isn't convenient to apply the discretization in the coordinate system placed in the middle point of the sonars arrangement (see fig. 20a). This approach causes that to compute r value the square root operation is needed. Much efficient approach consist in applying two discretization maps (see fig. 21) representing two look-up tables. They contain coordinates of the cell center expressed

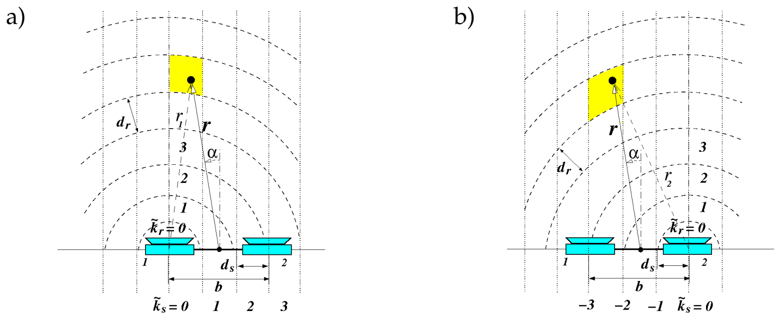


Fig. 21. Interpretation of parameters.

in the coordinates system of the middle of the sonars baseline. Executing two measurements the final result is the mean of values obtained from these two tables. The values r_1 and r_2 are directly obtained form measurements.

5.2 Basic error analysis

To estimate an computation error of the parameter k the formula (21) has to be considered. This formula shows that the parameter k is a function of r_{k-l}, r_{k-l-1} and d_s. Thus the error can be estimated as follows

$$\Delta k = \tfrac{1}{d_s^2}(r_{k-l}\Delta r_{k-l} + r_{k-l-1}\Delta r_{k-l-1}) + \tfrac{\Delta d_s}{d_s^3} \mid r_{k-l}^2 - r_{k-l-1}^2 \mid . \tag{23}$$

The interpretation of the symbols used in the formula is presented in fig. 22. It is shown in the context of a binaural sonar system. The form (23) indicates that the error can be reduced by

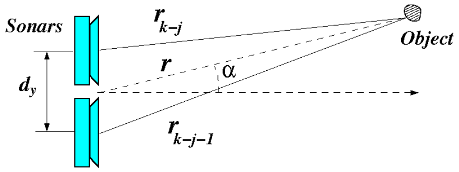


Fig. 22. The schema of sonars arrangement used to determine error values

enlarging the distance d_s between sonars. The chart in fig. 23 shows that the value of the error rapidly drops while d_s is enlarged up to about $9cm$. This conclusion isn't true with regard to a

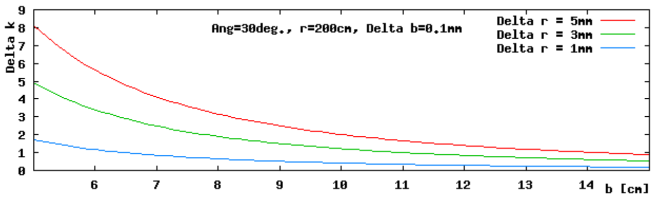


Fig. 23. The error estimation of k parameter as a function of the parameter b (distance between sonars). The error is estimated for an object placed at the distance $2m$ and the azimuth to the object is $30°$

real distance $y = kd_s$ because

$$\Delta kd_s = \frac{1}{d_s}(r_{k-l}\Delta r_{k-l} + r_{k-l-1}\Delta r_{k-l-1}) + \frac{\Delta d_s}{d_s^2} \mid r_{k-l}^2 - r_{k-l-1}^2 \mid . \tag{24}$$

The chart in fig. 24 illustrates this relation. Unfortunately it isn't possible to increase the distance between sonars significantly. If it is enlarge too much, an object cannot be detected by all sonars. It is due to a restricted size of the emission angle. Another reason is that sonars have much lower sensitivity when a signal is received from a direction being far from their acoustic axis. For Polaroid transducers the distance between sonars up to $8cm$ seems to be reasonable. The error estimations presented in fig. 23 and fig. 24 are computed for a specific direction. But the conclusions are true for the whole range of the sonar sensibility. This is due to the fact that the error almost doesn't depend on the direction. The second component of the form (23) has

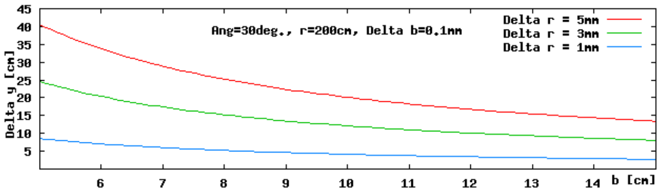


Fig. 24. The error estimation of the coordinate y of the object position. This error is a function of the parameter b (distance between sonars)

very small influence into the final value. This is also the reason that the error is almost linear in the sense of the distance to an object.

The same it can be said about the influence of the error Δr of distance measurement. Increasing the accuracy of the distance measurement it is possible to reduce the error of the parameter k. It can also be done by enlarging the distance d_s between sonars but unfortunately in a very limited range. Because it isn't possible to increase the distance between sonars without loosing detection of object located in the nearest robot surrounding. The error analyze presented in this section shows that it is possible if the accuracy of distance measurement is $1mm$ and the distance between sonars is $8cm$. In this way the condition $\Delta k \leq 1$ is guaranteed up to about $3m$ to an object.

6. Experiments

This section presents a sample of a few experiments. The obtained measurement data have been used for testing the presented approaches. The experiments checking the differentiation method have been performed using a binaural sonar system whose length of the baseline is $b = 13.6cm$. The same sonar arrangement was used for testing the triangulation method. The aforementioned system consists of Polaroid transducers 600–series model and the 6500 ranging modules. They were modified in the same way described in the section 2. The sonar system used in experiment isn't precise. Its measurement error is $2mm$. Therefore objects being the targets had to be placed enough close to the sonar system.

To test the method of objects differentiation presented in this chapter three objects were used i.e. a single plane, two planes arranged in a corner shape and others two planes arranged in an edge. They were placed at the distance about $0.5m$. The measurement were performed for three orientations. They were restricted to very narrow sector because echos were detected using the threshold method. Out of this sector the distance measurement is rapidly changed. It is due to attenuation of a received signal amplitude. Because the error of measurement was $2mm$, thus $\Delta C_e = 8mm$ and $\Delta C_w = \Delta C_c = 32000mm^2$. The computed values of C_e, C_w and

		edge			wall			corner		
	ΔC	$-3.7°$	$0°$	$3.7°$	$-3.7°$	$0°$	$3.7°$	$-3.7°$	$0°$	$3.7°$
C_e	8	**7**	**5**	**3**	24	24	20	-11	-16	-10
C_w	32000	-44503	-53583	-61575	**26180**	**26340**	**10436**	-117191	-139968	-115468
C_c	32000	102113	94313	85881	174076	173916	156604	**24945**	**4800**	**29908**

Table 1. The results of the object classification

C_c for all performed measurements are presented in tab. 1. They allow properly to classify all objects.
In the second part the previously obtained results of measurements have been used to determine the location of the objects. The computations have been performed for $\Delta r = 1.0625mm$ (it gives $b = 128\Delta r$), $d_r = 16\Delta r$, $n = 16$ and $m = 8$ (parameters meaning see (22) and fig. 21). Applying these parameters a look-up table has been constructed containing values of r and α in each cell. They represent the polar coordinates of the middle of the grid cell (see fig. 20b). Using the results of measurements the indexes of $\tilde{k}_y$ and $\tilde{k}_r$ have been computed. These indexes were used for r and $alpha$ values selection from the look-up tables. The procedure of this procedure has been described in subsectionsec-multi-sonar-system. The obtained results are presented in tab. 2. They are not very accurate due to measurements errors and errors

	edge			wall			corner		
	$-3.7°$	$0°$	$3.7°$	$-3.7°$	$0°$	$3.7°$	$-3.7°$	$0°$	$3.7°$
$\tilde{r}$ [mm]	515	510	510	520	515	520	515	535	535
$\tilde{\alpha}$ [°]	-2.9	-1.0	2.9	-3.0	1.0	4.8	-5.1	1.8	4.07

Table 2. The results of the objects localization

caused by discretization. But differences seems do be acceptable.

7. Summary

The approaches presented in this paper make possible to implement the methods using a simple micro-controller. It can be done because operations on integer numbers are only needed. For the method of object differentiation this feature is obtained by simplifying the form of the criterion. But the simplification doesn't mean lower precise in classification. For the object localization method the confinement to the integer number arithmetic is achieved by exploiting the discretization proposed in the paper. Unfortunately, it introduces additional source of errors. Considering the both methods the critical issue is the the error of TOF measurement. To be able to obtain a full implementation of the proposed method using low price hardware a proper echo detection method is needed. The easiest approach to this problem is a detection of exceeding a given threshold. But it is very sensitive to a signal amplitude and therefore introduce rather big errors. This disadvantage can be eliminated using an approach based on a correlation function. But it requires more computation power of the system and more complicated hardware. In this context the future work is concentrated on a method of the echo detection and the TOF measurement performance which should be able to combine simplicity of the threshold method and the accuracy of the methods based on the correlation function. Reaching satisfactory result it will be possible to build a low price smart sensor allowing to locate an object an identify some features of its surface shape. This seems to be a very good alternative solution comparing with traditional ultrasonic range-finders. Besides, it can be a very good complementary equipment to other set of sensors like laser range-finder, PSD sensors and vision systems.

8. References

Akbarally, H. & Kleeman, L. (1995). A sonar sensor for accurate 3d target localisation and classification. In: *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, volume 3, pages 3003–3008.

Angrisani, L. & Moriello, R. S. L. (2006). Estimating ultrasonic time-of-flight through quadrature demodulation. *IEEE Trans. on Instrumentation and Measurement*, 55(1):54 – 62.

Ayrulu, B.; Barshan, B., & Utete, S. (1997). Target identification with multiple logical sonars using evidential reasoning and simple majority voting. In: *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 3, pages 2063–2068.

Brown, M. K. (1985). Feature extraction techniques for recognizing solid objects with an ultrasonic range sensors. *IEEE Journal of Robotics and Automation*.

Cheeke, J. D. N. (2002). *Fundamentals and Applications of Ultrasonic Waves*. CRC Press.

Egaña, A.; Seco, F., & Ceres, R. (2008). Processing of ultrasonic echo envelopes for object location with nearby receivers. *IEEE Trans. on Instrumentation and Measurement*, 57(12).

Heale, A. & Kleeman, L. (2001). Fast target classification using sonar. In: *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 3, pages 1446–1451 vol.3.

Jimenez, J.; Mazo, M.; Urena, J.; Hernandez, A.; Alvarez, F.; Garcia, J., & Santiso, E. (2005). Using pca in time-of-flight vectors for reflector recognition and 3-d localization. *Robotics, IEEE Transactions on*, 21(5):909–924.

Kleeman, L. (2004). Advanced sonar with velocity compensation. *Int. J. Robotics Res.*, 23.

Kleeman, L. & Kuc, R. (1995). Mobile robot sonar for target localization and classification. *Int. J. Robotics Res.*, 14(4):295 – 318.

Kreczmer, B. (1998). Path planning system for car-like robot. In: *Proc. 1998 IEEE Int. Conf. Robot. Automat.*, volume 1, pages 40 – 45.

Kreczmer, B. (2006). 3d object localization and distinguishing using triangular-shaped ultrasonic system. In: *Proc. 12th IEEE Int. Conf. MMAR 2006*, pages 931 – 936.

Kuc, R. (2001). Pseudoamplitude scan sonar maps. *IEEE Trans. Robot. Automat.*, 17(5):787 – 770.

Kuc, R. (2007). Neuromorphic processing of moving sonar data for estimating passing range. *IEEE Sensors Journal*, 7(5):851 – 859.

Kuc, R. & Siegel, M. W. (1987). Physically-based simulation model for acoustic sensor robot navigation. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 9:766 – 778.

Leonard, J. J. & Durrant-Whyte, H. F. (1991). Mobile robot localization by tracking geometric beacons. *IEEE Trans. Robot. Automat.*, 7(3):376–382.

Li, H. M. & Kleeman, L. (1995). A low sample rate 3d sonar sensor for mobile robots. In: *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Confe rence on*, volume 3, pages 3015–3020.

Ochoa, A.; Urena, J.; Hernandez, A.; Mazo, M.; Jimenez, J., & Perez, M. (2009). Ultrasonic multitransducer system for classification and 3-D location of reflectors based on pca. *Instrumentation and Measurement, IEEE Transactions on*, 58(9):3031–3041.

Peremans, H.; Audenaert, K., & Campenhout, J. M. V. (1993). A high-resolution sensor based on tri-aural perception. *IEEE Trans. Robot. Automat.*, 9(1):36 – 48.

Schillebeeckx, F.; Reijniers, J., & Peremans, H. (2008). Probabilistic spectrum based azimuth estimation with a binaural robotic bat head. In: *4th Inter. Conf. on Autonomic and Autonomous Systems*, pages 142 – 147.

Heading Measurements for Indoor Mobile Robots with Minimized Drift using a MEMS Gyroscopes

Sung Kyung Hong and Young-sun Ryuh
Dept. of Aerospace Engineering, Sejong University
Korea
Center for the Advanced Robot Industry,
Korea Institute of Industrial Technology
Korea

1. Introduction

Autonomous cleaning robots increasingly gain popularities for saving time and reduce household labor. Less expensive self-localization of the robot is one of the most important problems in popularizing service robot products. However, trade-offs exist between making less expensive self-localization systems and the quality at which they perform. Relative localizations that utilize low-cost gyroscope (hereafter refer to as a gyro) based on technology referred to as Micro Electro-Mechanical System (MEMS) with odometry sensors have emerged as standalone, and robust to environment changes (Barshan & Durrant-Whyte, 1995; De Cecco, 2003; Jetto et al., 1995). However, the performance characteristics associated with these low-cost MEMS gyros are limited by various error sources (Yazdi et al., 1998; Chung et al., 2001; Hong, 1999). These errors are subdivided into two main categories according to their spectral signature: errors that change rather rapidly (short-term) and rather slowly (long-term) (Skaloud et al., 1999). Due to these error sources, the accumulated angular error grows considerably over time and provides a fundamental limitation to any angle measurement that relies solely on integration of rate signals from gyros (Hong, 2003; Stilwell et al., 2001).
Currently, because of their potential for unbounded growth of errors, odometry and gyro can only be used in conjunction with external sensors that provide periodic absolute position/attitude updates (Barshan & Durrant-Whyte, 1995; De Cecco, 2003; Jetto et al., 1995). However, such aiding localization schemes not only increase installation and operating costs for the whole system, but also only lead to compensation of the long-term inertial errors. That is, the angle deterioration due to quickly changing errors (system noise, vibration) is not detectable by external sensors over a short time period (Sun et al., 2005; Hong, 2008a). Considering this concept, better performance of a localization system can be expected if short-term noise in the gyro signal is suppressed prior to integration. In the previous study (Hong, 2008a; Hong & Park, 2008b), it has been shown that the threshold

filter is very effective for suppression the broadband noise components at the gyro output. However, the threshold filter was partially effective only when there's no turning motion.
To improve our previous studies and achieve further minimized drift on yaw angle measurements, this chapter examines a simple, yet very effective filtering method that suppresses short-term broadband noise in low cost MEMS gyros. The main idea of the proposed approach is consists of two phases; 1) threshold filter for translational motions, and 2) moving average filter for rotational motions to reject the broadband noise component that affects short-term performance. The switching criteria for the proposed filters are determined by the motion patterns recognized via the signals from the gyro and encoder. Through this strategy, short-term errors at the gyro signal can be suppressed, and accurate yaw angles can be estimated regardless to motion status of mobile robots. This method can be extended to increase the travel distance in-between absolute position updates, and thus results in lower installation and operating costs for the whole system.
An Epson XV3500 MEMS gyro (Angular Rate Sensor XV3500CB Data sheets, 2006) was selected as the candidates at our lab. It has performance indexes of 35°/hr bias stability, 3.7°/√hr ARW(Angle Random Walk). These specifications are saying that, if we integrate the signals of this gyro for 30 minutes (assumed cleaning robot's operating time) it can be supposed to have the standard deviation of the distribution of the angle drift up to 20°, which provides a fundamental limitation to any angle measurement that relies solely on integration of rate. However, experimental results with the proposed filtering method applied to XV3500 demonstrate that it effectively yields minimal-drift angle measurements getting over major error sources that affect short-term performance.
This chapter is organized as follows. Section 2 describes the error characteristics for the gyro XV3500. In Section 3, the algorithms for minimal-drift heading angle measurement including self-identification of calibration coefficients, threshold filter, and moving average filter are presented. In Section 4, the experimental results are provided to demonstrate the effectiveness of the proposed method. Concluding remarks are given in Section 5.

2. The error characteristics of a MEMS gyro

In general, gyros can be classified into three different categories based on their performance: inertial-grade, tactical-grade, and rate-grade devices. Table 1 summarizes the major requirements for each of these categorizes (Barbour & Schmidt, 2001).
The MEMS gyros, Epson XV3500, used in this research are of a quality that is labeled as "rate grade" devices. This term is used to describe these sensors because their primary application is in the automotive industry where they are used for active suspension and skid control via rate measurement, not angle measurement. Therefore it is normally considered that to estimate precise angle with unaided (without odometer/velocity or GPS or magnetometer aiding) rate-graded MEMS gyro is a quite challenging problem.
These sensors range in cost from $25 to $1000 and are expected to drop in price in the future. In this study the rate gyro that was used and tested extensively was the Epson XV3500 (shown in Figure 1) that costs less than $50.
The output of gyro has rather complex noise characteristics that are produced by many different error sources. A detailed general model and a thorough discussion on each individual error component can be found for instance in (Titterton & Weston, 2004) and references therein.

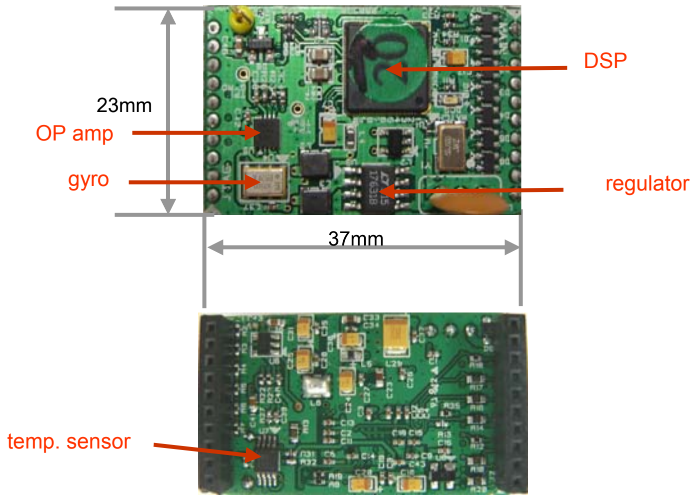


Fig. 1. Epson XV3500 gyro module

	Rate-grade	Tactical-grade	Inertial-grade
Angle Random Walk (°/√hr)	>0.5	0.5~0.05	<0.001
Bias Drift (°/hr)	10~1,000	0.1~10	<0.01
Scale Factor Accuracy (%)	0.1~1	0.01~0.1	<0.001

Table 1. Performance requirement for different type of gyros

In the following, as shown in Figure 2, gyro errors are subdivided into two main categories according to their spectral signature: errors that change rather rapidly (short-term) and rather slowly (long-term). The characteristics of two different categories can be analyzed through an Allan-variance chart. Figure 3 shows an Allan-variance chart for XV3500s. This chart is representative of all the other XV3500 gyros tested.

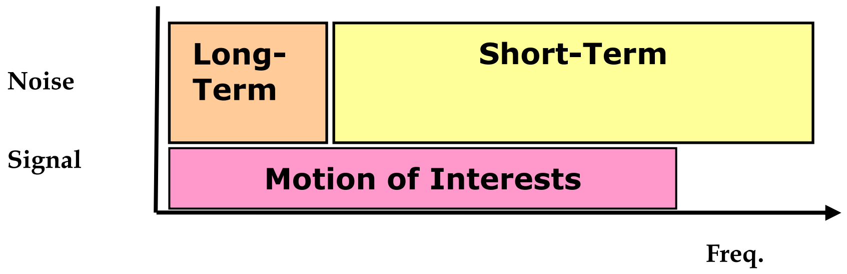


Fig. 2. A schematic plot of gyro signal in the frequency domain

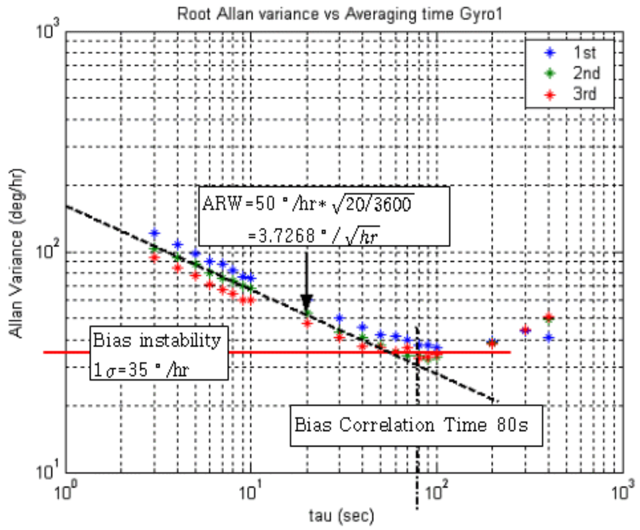


Fig. 3. Allan Variance Chart

2.1 Long-Term Errors

This category combines errors in the low frequency range. Individual error sources belonging to this category would include the main components of the gyro bias drift, scale factor, misalignment errors, and so on. From Figure 3, it is shown that the gyros exhibit a long-term instability, what is called bias instability, which tends to dominate the output error after about 200 sec. The initial upward slope of approximately +1/2 indicates that the output error in that time period is predominately driven by an exponentially correlated process with a time constant much larger than 200 sec. In Figure 3, it is shown that the mean bias instability, which is the maximum deviation of the random variation of the bias, is 35°/hr. This instability tends to dominate the long-term performance. Usually, updating gyro with some other angle reference observation or self-calibration prevents divergence due to such long-term errors.

2.2 Short-Term Errors

This category complements the long-term inertial error spectrum up to half of the gyro sampling rate. The prevailing error sources include Angle Random walk (ARW) defined as the broadband sensor noise component, and correlated noise due to vehicle vibrations. Without some other angle reference, this will be a fundamental uncertainty in the result of the angle calculation. Better performance of a system can be expected if short-term noise in the gyro is suppressed prior to integration. To a certain extent, this can be achieved by implementing optimal low pass filter beyond motion bandwidth. After two repeated experiments for 10 sets of gyros, averaged ARW was 3.7°/√hr as shown in Figure 3. This represents the short-term performance limit of XV3500 in that the standard deviation of the angle distribution obtained by integrating rate signals for an hour is 3.7 degrees

In short, we found that XV3500 as a low-cost MEMS gyro used here is dominated by three major error sources: scale-factor error (s) and bias (b) that affect long-term performance, and ARW defined as the broadband noise component (w) that affects short-term performance. We assume the output of the gyro is written

$$r_m(t) = (1+s)r(t) + b + w \tag{1}$$

where $r_m(t)$ is the measured rate from gyro and $r(t)$ is the true rate about the sensitive axis.

3. Minimal-drift yaw angle measurement

For long-term errors, as conceptually described in Figure 4, it can be reduced by periodical calibration and compensation of scale-factor error (s) and bias (b) (Hong, 2008; Hong & Park, 2008). In this chapter, a novel self-calibration algorithm is suggested for improved long-term performance.
For short-term errors (ARW and vibration), most of the signal processing methodologies currently employed utilize Kalman filtering techniques to reduce the short-term error and to improve the estimation accuracy. Unfortunately, the convergence of these methods (the time to remove the effect of the short-term error and to provide an accurate estimate) during the alignment processes can take up to 15min. In addition to their time-consuming algorithms, these techniques are complex to design and require a considerable effort to achieve real-time processing. Although some modern techniques have been introduced to reduce the noise level and some promising simulation results have been presented, real-time implementation with an actual gyro has not yet been reported.
To achieve further minimized drift on yaw angle measurements, in this section, we examine a simple, yet very effective filtering method that suppresses short-term broadband noise in low cost MEMS gyros. The concept of the proposed strategy is described in Figure 4. The main idea of the proposed approach is consists of two phases; 1) threshold filter for translational motions, and 2) moving average filter for rotational motions to reject the broadband noise component (w) that affects short-term performance. The switching criteria for the proposed filters are determined by the motion patterns of mobile robots.

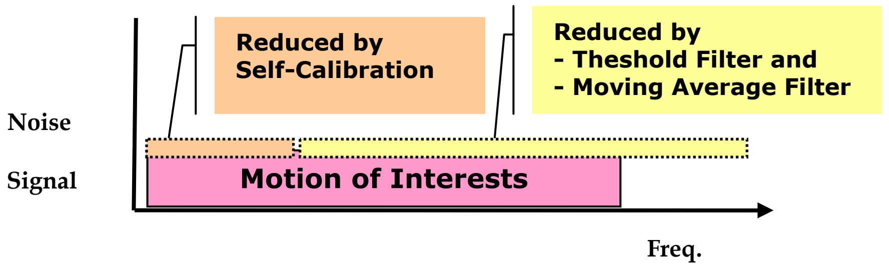


Fig. 4. The concept of de-noising

3.1 Self-calibration with least square algorithm

Our initial focus is on self-calibration of the gyro, that is, to update calibration coefficients (s and b) that have been changed somehow (temperature change, and aging, etc). During startup phase of cleaning robot, time-varying calibration coefficients of both s and b are simultaneously estimated and stored. Subsequently, when a gyro measurement is taken, it is compensated with the estimated coefficients. This means that the calibration coefficients that affect long-term performance are updated regularly so that the performance is kept consistent regardless of unpredictable changes due to aging. Ignoring the broadband noise component (w) in (1), the output of the gyro can be written

$$r_m(t) = (1+s)r(t) + b \tag{2}$$

We will often find it more convenient to express (2) as

$$r(t) = \bar{s}\, r_m(t) + \bar{b} \tag{3}$$

where $\bar{s} = \dfrac{1}{1+s}$, and . $\bar{b} = \dfrac{-b}{1+s}$.

1) Least square algorithm with heading reference

The least square algorithm presented in (Hong, 1999; Hong, 2008) is introduced to find the calibration coefficients, $\bar{s}$ and $\bar{b}$ of (3). Consider the discrete-time state equation relating yaw angle and yaw rate

$$\psi(k+1) = \psi(k) + h\,(\bar{s}\, r_m(k) + \bar{b}),\ \ k \geq 0 \tag{4}$$

where $\psi(k+1)$ is the future yaw angle, $\psi(k)$ is the initial yaw angle, and h is sample period. For any future index $n > k \geq 0$, (4) is rewritten

$$\psi(k+n) = \psi(k) + h\,\bar{s} \sum_{i=k}^{k+n-1} r_m(i) + n\,h\,\bar{b} \tag{5}$$

Taking a number of reference (true) values of yaw angle from the predetermined known motion profile (i.e., typical open-loop controlled motion profile that is assumed to be lifetime identical with precise data set provided by a factory) of robot and measurements from the gyro for increasing n and stacking the equations yields the matrix equation:

$$z = Gq \tag{6}$$

where

$$z = \begin{bmatrix} \psi(k+1) - \psi(k) \\ \psi(k+2) - \psi(k) \\ \vdots \\ \psi(k+n) - \psi(k) \end{bmatrix},\ q = \begin{bmatrix} \bar{s} \\ \bar{b} \end{bmatrix}, G = [G_1 \quad G_2]$$

and

$$G_1 = \begin{bmatrix} h\,r_m(k) \\ h\sum_{i=k}^{k+1} r_m(i) \\ \vdots \\ h\sum_{i=k}^{k+n-1} r_m(i) \end{bmatrix}, \; G_2 = \begin{bmatrix} h \\ 2h \\ \vdots \\ nh \end{bmatrix}$$

Now a least squares estimate of the scale and bias coefficients can be found by solving (6) for q

$$q = (G^T G)^{-1} G^T z \tag{7}$$

The vector q can be found as long as $G^T G$ is nonsingular, meaning that the robot should be changing rate during the calibration.

2) Evaluation

To evaluate the performance of this algorithm, some simulations are done with yaw angle reference data for some specified motion profiles. For all simulations, gyro scale and bias factors were set to s=-0.1 rad/sec/volts and b=0.1 rad/sec, corresponding to coefficient value of $\bar{s}$ =1.111, $\bar{b}$ =-0.111. Figure 5 shows the discrepancies between reference angle and gyro angle (integration of rate). The proposed least square algorithm found the coefficients of s and b with error of 4.8% and 20.26%, respectively. Subsequently, when the gyro measurements are compensated with the estimated coefficients, they show almost identical to reference data as shown in Figure 5.

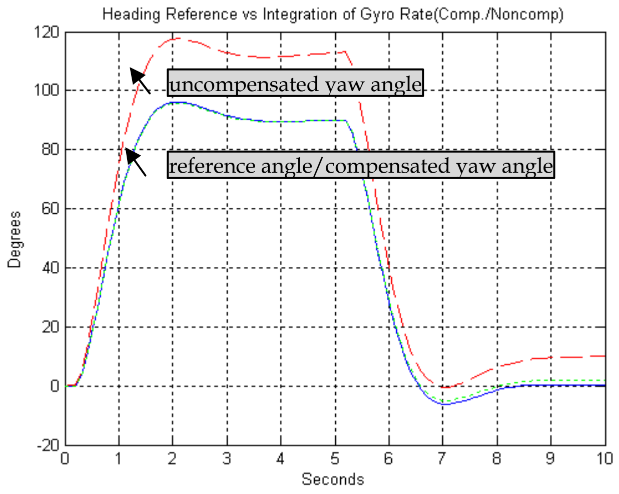


Fig. 5. Self-calibration result with least square algorithm

3.2 Identification of motion type

Before applying two different filtering schemes for short-term performance, we have to identify types of vehicle maneuver (standing/straight, steady turning, or transient). Figure 6 shows typical experimental data of XV3500 for different maneuvers. Based on the characteristics of sensor responses, the each type of vehicle maneuver is identified as:

1) If the absolute value of gyro signal lies under a certain value (with the current setting < 0.3 deg/s), the vehicle maneuver is identified as standing or straight path.
2) On the other hand, if the vehicle maneuver is not standing or straight path, and the absolute value of the error between gyro and encoder is less than a certain value (with the current setting < 5 deg/s), the vehicle maneuver is identified as steady turning.
3) If not both of maneuvers, it is considered as transient and no filter is applied to retain given edge sharpness.

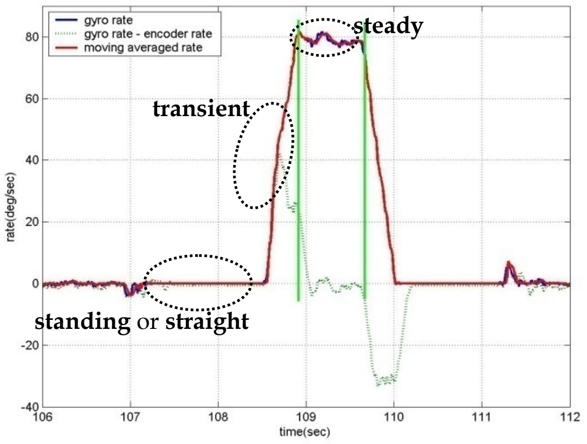


Fig. 6. Identification of typical vehicle maneuvers

3.3 Threshold filter

If the scale factor error (s) and bias (b) of the gyro are identified and compensated with the self-calibration procedures described in the section 3.1, the output of the gyro can be written as

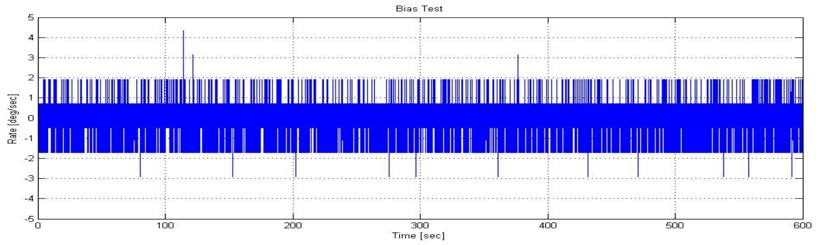
$$r_m(t) = r(t) + d(t) + w \tag{8}$$

where d is the residual uncompensated error of the scale factor and bias. The dominant part of error d comes from the stochastic bias which is the unpredictable time-varying drift of bias, and the remaining part is the effect of uncompensated scale factor error which can be included as one source of the stochastic bias. Typically, d is modeled as a random walk as following.

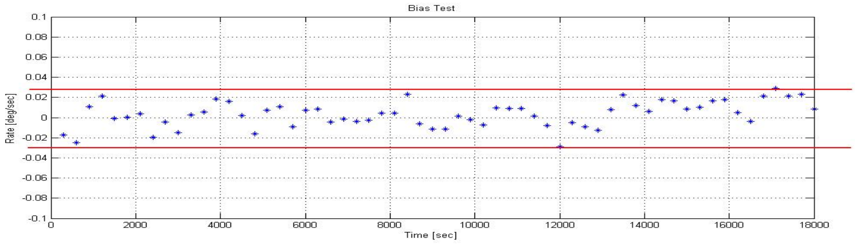
$$\dot{d} = n \tag{9}$$

where *n* is zero-mean white noise with the spectral density *Q* which can be determined by analyzing the gyro output.

Figure 7(a) shows a typical output of EPSON XV3500 gyro. Note that the peak-to-peak noise is about 2 *deg/sec* dominated by the broadband noise (*w*). The residual error (*d*) is shown in Figure 7(b), which is obtained by averaging the gyro output based on an averaging time 5 min. after collecting a long term sequence of data. In this figure, the upper and lower bounds of the uncertainty are also plotted. These uncertainty bounds limit the minimum detectable signal of angular rate, and thus limit the accuracy of the heading angle calculated by integrating angular rate.



(a)



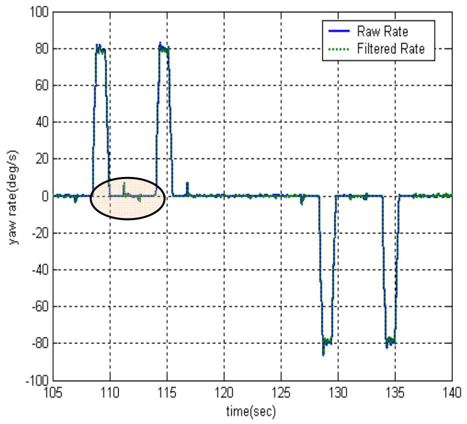
(b)

Fig. 7. Typical output of XV3500 (a) broad band noise(*w*), (b) residual error (*d*)

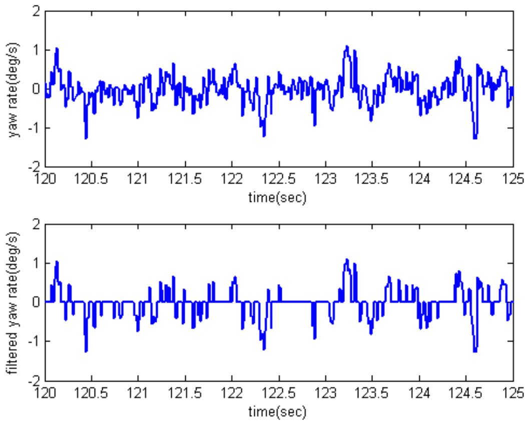
For the standing or straight maneuver, we implemented threshold filter. That is, the values in the gyro signal which lie under a certain threshold are filtered out and set to zero.

$$if\ |r_m| < r_{thres},\ then\ r_m = 0 \tag{10}$$

This means that the effects of ARW or vibration are ideally eliminated when there's no turning motion. This filters out noise but of course also eliminates the gyro's capability to sense very slow turns (with the current setting < 0.3 deg/s). This has only limited significance, however, since the cleaning robot which is the application target of this study can only move with a certain minimum angular velocity (>>1 deg/s). Of course, slow changes in direction during "pure" straight path maneuver are also ignored but this is not supposed to be a major error source in the vehicle's odometry. However, it should be noted that this filter is applicable only when there's no turning motion. Figure 8(a) shows some part of rate signals captured at typical maneuvers of the platform, and the effectiveness of proposed threshold filter can be seen in Figure 8 (b), that is zoomed in for the circled region (straight maneuvering state) of Figure 8(a).



(a)



(b)

Fig. 8. Raw and filtered yaw rate signals for the straight maneuvering state

3.4 Recursive moving average filter

The moving average is the most common filter, mainly because it is the easiest digital filter to understand and use. As the name implies, the moving average filter operates by averaging a number of points from the input signal to produce each point in the output signal. As the number of points in the filter increases, the noise becomes lower; however, the edges becoming less sharp. In this study, for the optimal solution that provides the lowest noise possible for a given edge sharpness, the moving average filter is applied only when the vehicle maneuvers steady turning. Thus the given sharpness of transient motion could be retained without distortion.

For computational efficiency we perform the calculation of the mean in a recursive fashion. A recursive solution is one which depends on a previously calculated value. Suppose that at any instant k, the average of the latest n samples of a data sequence, x_i, is given by:

$$\bar{x}_k = \frac{1}{n} \sum_{i=k-n+1}^{k} x_i \tag{11}$$

Similarly, at the previous time instant, k-1, the average of the latest n samples is:

$$\bar{x}_{k-1} = \frac{1}{n} \sum_{i=k-n}^{k-1} x_i \tag{12}$$

Therefore,

$$\bar{x}_k - \bar{x}_{k-1} = \frac{1}{n} \left[\sum_{i=k-n+1}^{k} x_i - \sum_{i=k-n}^{k-1} x_i \right] = \frac{1}{n} \left[x_k - x_{k-n} \right] \tag{13}$$

which on rearrangement gives:

$$\bar{x}_k = \bar{x}_{k-1} + \frac{1}{n} \left[x_k - x_{k-n} \right] \tag{14}$$

This is known as a moving average because the average at each kth instant is based on the most recent set of n values. In other words, at any instant, a moving window of n (current setting in this study is 10) values are used to calculate the average of the data sequence (see Figure 9).

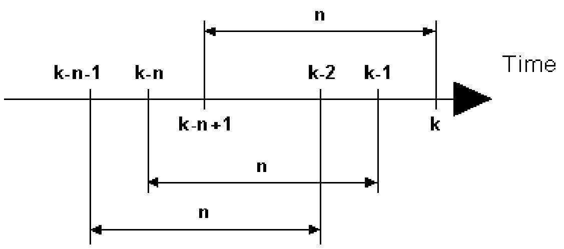


Fig. 9. Moving Window of n Data Points

4. Experimental Results

4.1 Experimental Setup

The practical technique proposed in this chapter was tested using an Epson XV3500, a low-cost MEMS gyro with 35 deg/h bias drift, and 3.7°/√hr ARW. It is mounted on the robot platform that is a self-made conventional two-wheel differential drive with two casters (Figure 10). The robot was tested on a preprogrammed trajectory for 320 sec and self-calibration period are given for 20sec. We used the Hawk Digital system from Motion Analysis, Inc. to obtain the ground truth data of the robot heading. The robot traveled at a maximum speed of 2.0 m/s, and the reference data and XV3500 gyro measurements were acquired at sampling rates of 20Hz and 100Hz, respectively.

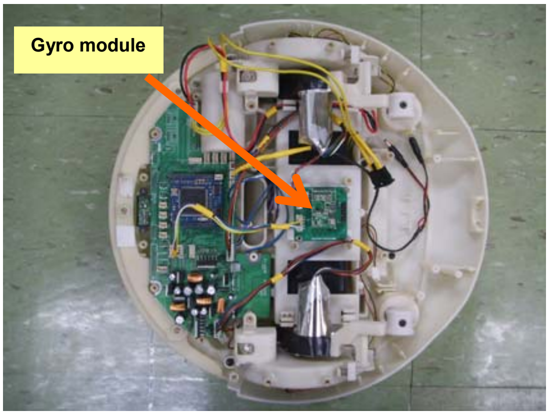


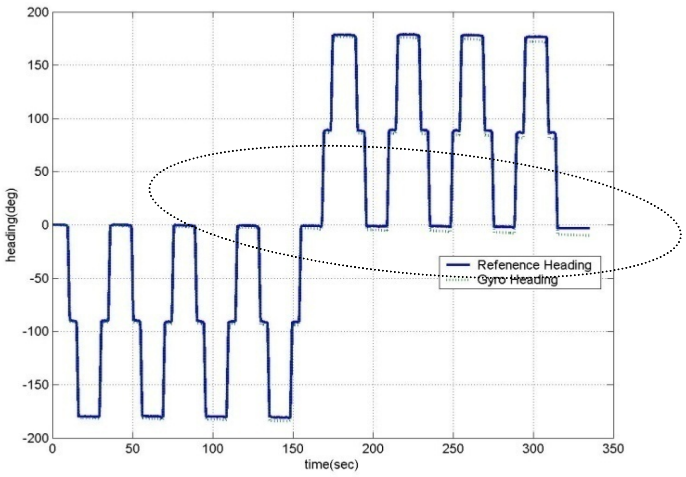
Fig. 10. Robot Platform

4.2 Results

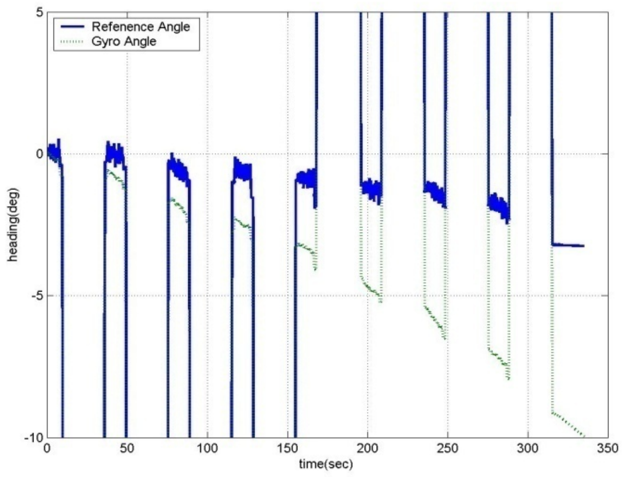
The yaw angles correspond to preprogrammed trajectory is shown in Figure 11(a). The angle error is the mean absolute value of error between the true and the estimated value. From Figure 11(b), that is exaggerated some part of Figure 8(a), it can be seen that the error of gyro angle (pure integration of rate signal) is growing over time. It shows a fundamental limitation to any angle measurement that relies solely on pure integration of rate signal. On the other hand, after self-calibration or threshold filtering, the mean error has been reduced by 48% and 59% respectively, compared with pure integration result (shown in Figure 11(c)). This improvement, of course, is due to proper handling of major error sources that affects long-term or short-term performance when straight path maneuvering, respectively. However, both results show some drift over time. From Figure 11(d), it can be seen that the heading angle of the proposed method 1 (both self-calibration and threshold filtering) follows the true yaw angle showing little drift at all time, showing 64% error reduction. This is considered to be quite good performance for automotive grade gyros. However, it should be noted that perfect drift free heading is possible only when ARW that exists even in the turning motion is rejected. In our proposed method 1, threshold filter can reject noise components only when no rotation is applied. Figure 11(e) shows proposed method 2 that combines both threshold filter for straight maneuver and moving average filter for steady turning maneuver. It almost follows the true heading showing little drift at all time, showing 74% error reduction. All the results are summarized in Table 1 with improved bias drift estimation.

	Mean Error (deg)	Standard Deviation(deg)	Estimated Bias Drift (deg/hr)
Pure Integration	3.2645	1.8838	36
Self-Calibration	1.6774	1.7599	18
Threshold filter	1.3503	1.7144	15
Proposed 1	1.2052	1.7745	13
Proposed 2	0.8353	1.9214	9

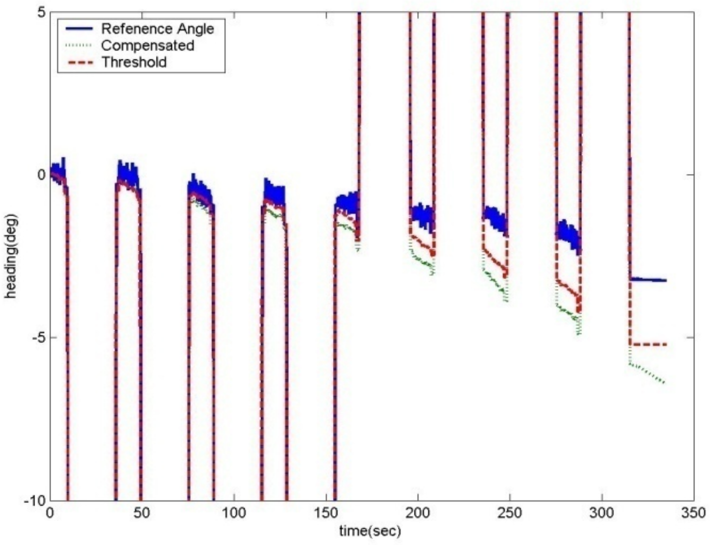
Table 2. Mean yaw angle for each method (deg)



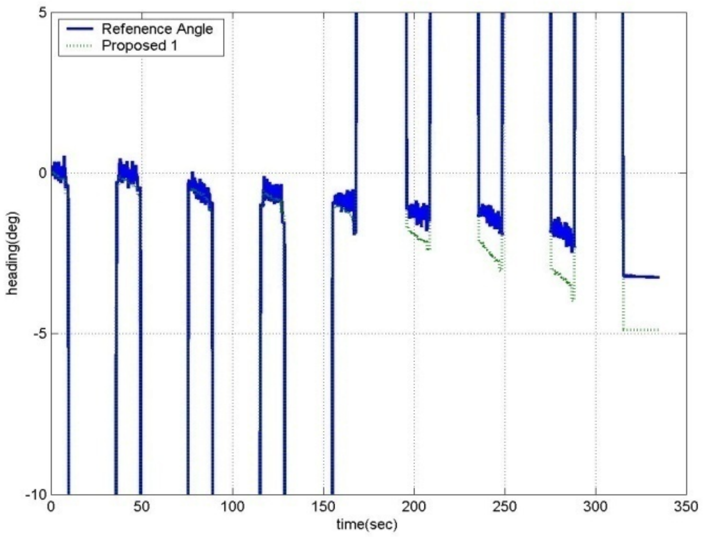
(a)



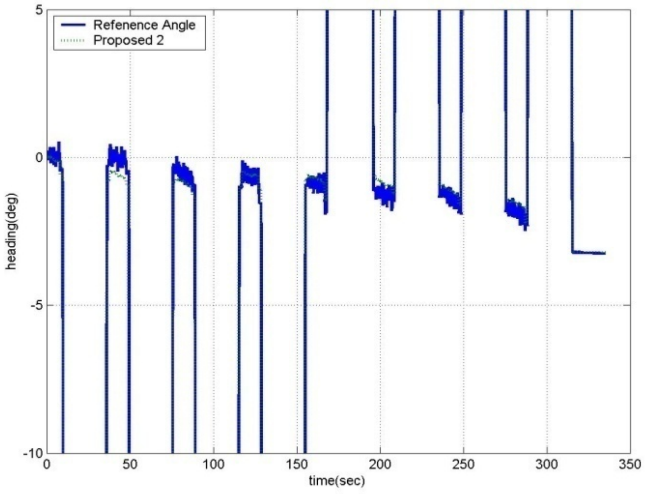
(b)


Refenence Angle
Compensated
Threshold
heading(deg)
time(sec)
5
0
-5
-10
0
50
100
150
200
250
300
350

(c)


Refenence Angle
Proposed 1
heading(deg)
time(sec)
5
0
-5
-10
0
50
100
150
200
250
300
350

(d)



(e)

Fig. 11. Experimental results for each method

5. CONCLUSION

To achieve further minimized drift on yaw angle measurements for mobile robot, this chapter examines a simple, yet a very practical and effective filtering method that suppresses short-term broadband noise in low cost MEMS gyros. This filtering method is applied after the self-calibration procedure that is a novel algorithm developed for the long-term performance. The main idea of the proposed approach is consists of two phases; 1) threshold filter for steady or straight path maneuver, and 2) moving average filter for steady turning maneuver. However, it should be noted that no filter is applied to retain given edge sharpness for transient period. The switching criteria for the proposed filters are determined by the motion patterns of vehicle recognized via the signals from both gyro and encoder. Through this strategy, short-term errors at the gyro signal can be suppressed, and accurate yaw angles can be estimated regardless of motion status.

The proposed method that combines both threshold filter for straight maneuver and moving average filter for steady turning maneuver was evaluated through experiments with Epson XV3500 gyro showing little drift at all time (estimated bias drift as 9 deg/hr), and 74% error reduction, compared with pure integration result. This method can be extended to increase the travel distance in-between absolute position updates, and thus results in lower installation and operating costs for the whole system.

6. Acknowledgement

This work was supported by National Strategic R&D Program for Industrial Technology from Ministry of Knowledge Economy, Korea. The contents of this chapter are reorgarnized based on our previous works (Hong & Park, 2008, Hong, Moon, & Ryuh, 2009)

7. References

Epson Toyocom Corp. (2006). Angular Rate Sensor XV3500CB Data sheets, Ver. 1.0, 2006

Barbour, N. & G. Schmidt (2001). Inertial Sensor Technology Trends, *IEEE Sensors Journal*, Vol. 1, No. 4, pp.332-339

Barshan, B. & Durrant-Whyte, H. F.(1995). Inertial Navigation System for Mobile Robots, *IEEE Transactions on Robotics and Automation*, Vol. 11, No. 3, pp.328-342

Chung, H.; Ojeda, L. & Borenstein, J. (2001). Accurate Mobile Robot Dead-reckoning with a Precision-calibrated Fiber Optic Gyroscope, *IEEE Transactions on Robotics and Automation*, Vol. 17, No. 1, pp.80-84

De Cecco, M.(2003). Sensor fusion of inertial-odometric navigation as a function of the actual maneuvers of autonomous guided vehicles, *Measurement Science and Technology*, Vol. 14, pp. 643-653

Hong, S. K. (1999). Compensation of Nonlinear Thermal Bias Drift of Resonant Rate Sensor (RRS) using Fuzzy Logic, *Sensors and Actuators A-Physical*, Vol. 78, No. 2, pp.143-148

Hong, S. K. (2003). Fuzzy Logic based Closed-Loop Strapdown Attitude System for Unmanned Aerial Vehicle (UAV), *Sensors and Actuators A-Physical*, Vol. 107, No. 1, pp.109-118,

Hong, S. K. (2008a). A Fuzzy Logic based Performance Augmentation of MEMS Gyroscope, *Journal of Intelligent & Fuzzy Systems*, Vol. 19, No. 6, pp. 393-398,

Hong, S. K. & Park, S. (2008b). Minimal-Drift Heading Measurement using a MEMS Gyro for Indoor Mobile Robots, *Sensors Journal*, Vol. 8, No. 11, pp. 7287-7299

Hong, S. K.; Moon, S. & Ryuh, Y. (2009). Angle Measurements for Mobile Robots with Filtering of Short-Term in MEMS gyroscopes, in Press *Transactions of the Institute of Measurement and Control.*

Jetto, L.; Longhi, S. & Vitali, D. (1999). Localization of a wheeled mobile robot by sensor data fusion based on a fuzzy logic adapted Kalman filter, *Control Engineering Practice*, Vol. 7, pp. 763-771

Skaloud, J.; Bruton, A. M. & Schwarz, K. P. (1999). Detection and Filtering of Short-Term noise in Inertial Sensors, *Journal of the Institute of Navigation*, Vol. 46, No. 2, pp. 97-107

Stilwell, D.J.; Wick, C.E. & Bishop, B.E. (2001). Small Inertial Sensors for a Miniature Autonomous Underwater Vehicle, *Proc. of IEEE International Conference on Control Applications,* pp. 841—846, Sept. 2001 , Mexico

Sun, F.; Luo, C. & Nie, Q. (2005). Research on Modeling and Compensation Method of Fiber Optic Gyro' Random Error, *Proc. of IEEE Conference on Mechatronics and Automation*, pp. 461—465, July 2005, Canada

Titterton, D. H. & Weston, J.L. (2004). Strapdown inertial navigation technology-2nd Edition, IEE Radar, Sonar, Navigation and Avionics Series 17,

Yazdi, N.; Ayazi, F. & Najafi, K. (1998). Micromachined Inertial Sensors, *Proc. IEEE*, pp.1640-1659

Methods for Wheel Slip and Sinkage Estimation in Mobile Robots

Giulio Reina
University of Salento
Italy

1. Introduction

For autonomous mobile robots driving across soft soils, such as sand, loose dirt, or snow, it is important to deal with the dynamic effects occurring at the wheel-terrain interface. The most prevalent of these effects are wheel slip and sinkage, which greatly affect a robot's mobility. In general, a vehicle traversing loose sand might experience substantial slip and sinkage, and, hence, poor mobility (see Fig. 1). Conversely, a robot traversing firm clay might undergo small slip and sinkage, and better mobility. As a notable example, field trials performed at the Jet Propulsion Lab (JPL), using a terrestrial analog of the Mars Exploration Rovers have confirmed that there is a great amount of slippage in the drive wheels during traversal of Mars-like terrain (Lindemann & Voorhees, 2005).
Nevertheless, conventional control and localization algorithms generally do not consider the physical characteristics of the vehicle and of its environment. Failure to understand these characteristics could lead to poor position estimation, poor traction, and possibly even to complete immobility. Specifically, the presence of wheel slip precludes the use of conventional dead-reckoning techniques for navigation (Baumgartner et al., 1992; Fuke & Krotkov, 1996; Ojeda et al., 2004), since they are based on the assumption that wheel revolutions can be translated into linear displacement relative to the ground. If one wheel slips, then the associated encoder will register wheel revolutions even though these revolutions do not correspond to a linear displacement of the wheel. Conversely, if one wheel skids, fewer encoder pulses will be produced. Additionally, in presence of side forces, the robot also experiences lateral slip that results in a gradual deviation of the vehicle from the intended path, and possibly in directional instabilities. Minimizing longitudinal and lateral slippage not only limits odometric errors but also reduces the overall energy consumption, and increases the robot's traction and climbing performance.
Wheel sinkage is also a key variable in estimating vehicle-terrain interaction. Wheels can sink in soft soils to depths sufficient to prohibit further motion. Sinkage measurements are as well as valuable for reducing position estimation errors. The accuracy of kinematic models used to estimate rover position updates depends on precise knowledge of the wheel radius, which is used to compute the equivalent linear distance travelled by a wheel from the encoder reading. As the load-bearing strength of soil varies, so does the amount of sinkage and the wheel effective rolling radius, thus decreasing the odometric accuracy.

To reduce the effect of propagating this non-systematic error during rover traverses on varied terrain, a means to measure wheel sinkage (and therefore effective wheel radius) is necessary. In addition, wheel sinkage has been shown to be an important input to terrain identification according to classical terramechanics theory (Iagnemma & Dubowsky, 2004). In summary, it is generally desirable to have the capability to sense excessive wheel slippage and sinkage so that corrective controls may be executed before the vehicle becomes immobile. In turn, the knowledge of the extent of wheel slip and sinkage can improve the accuracy of odometry-based localization system and provide important information about terrain properties.

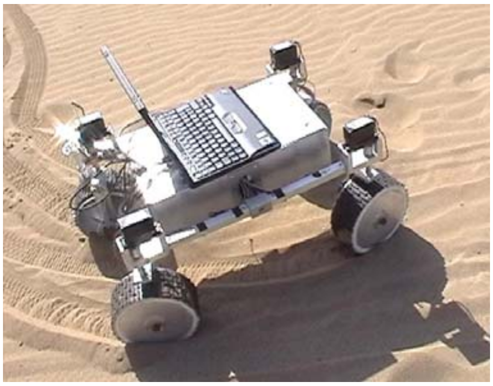


Fig. 1. Significant wheel slip and sinkage of the rover Dune, constructed at the University of Salento

In this chapter, methods for wheel slippage and sinkage detection are described, aimed to improve vehicle mobility on soft deformable terrain. In detail, a set of indicators to detect longitudinal slippage is described, based on the observation of purely proprioceptive sensors, such as wheel encoders, inertial measurement unit (IMU), and electrical current sensors. A novel approach for lateral slip estimation is also presented, which uses a rearward-facing video camera to measure the pose of the trace that is produced on the ground by the wheels, and to detect whether the robot follows the desired path or deviates from it because of slippage. Since this approach is based on Hough transform supported by Fuzzy logic to provide robust and accurate tracking of the wheel Trace, it is named “FTrace”. Finally, an algorithm for visual estimation of wheel sinkage is discussed. It assumes the presence of a monocular camera mounted to the wheel assembly, with a field of view containing the wheel-terrain interface. An artificial pattern, composed of concentric circumferences equally spaced apart on a white background, is attached to the wheel side in order to determine the contact angle with the terrain, following an edge detection strategy. Test results are presented showing the performance of the proposed methods to estimate wheel slippage and sinkage of rover-type robots moving across uneven terrain.

2.1 Related Work

Longitudinal slip can be estimated through the use of encoders by comparing the speed of driven wheels to that of undriven wheels on asphalt (Gustafsson, 1997); however this is not suitable for deformable surfaces and does not apply for all-wheel drive vehicles or those without redundant encoders. Recent research has investigated the use of either exteroceptive or absolute sensors, or both. Helmick et al., 2006, proposed a system for detecting slippage in a Mars rover based on fusing visual odometry and inertial measurements through a Kalman filter pose estimator. Baumgartner et al., 2001, utilized a sun sensor to improve rover state estimates. Combination of visual odometry with an absolute heading sensor, such as a compass or sun sensor, was shown to be effective for robust long-range navigation (Olson et al., 2001). In contrast to these methods, the approach described in this chapter for longitudinal slip detection relies on purely proprioceptive sensors.

In presence of side forces, the robot moves at an angle (i.e. the slip angle) with respect to its longitudinal axis, resulting in lateral slip as well. A large body of research work exists in the automotive community related to traction control, anti-lock braking systems (ABS), and electronic stability program (ESP). However, these works generally apply to asphalt roads and at significantly higher speeds than those typical for autonomous robots (Robert Bosch GmbH, 2007).

In principle, lateral drift can be estimated using visual motion estimation. It is possible to refer generally to two broad categories: landmark-based and optic flow-based methods. The former methods recognize either natural or artificial landmarks in the environment and, then, infer the position of the robot, usually, by triangulation. The latter approaches estimate the differential of successive images extracting optical flow vectors, which allow changes in the robot pose to be evaluated. Notable examples of this category include visual odometry methods, which were developed and demonstrated for rough-terrain robots (Konolige et al., 2007), and planetary rovers (Maimone et al., 2007). The purpose of visual odometry is that of incrementally estimating the motion of a mobile robot, by detecting and tracking interesting point-features of the environment over subsequent images, generally within a statistical framework for estimation. In some cases, the search for good features to track may be optimized selecting features on specific parts of the scene. Interestingly, point-features extracted from wheel tracks were used by the Mars exploration rover Opportunity for successful visual odometry on piles of sand at Meridiani Planum (see Section 5.1 in (Maimone et al., 2007)). In this regard, the FTrace system represents a different approach. The similarities and differences with respect to existing literature can be summarized in the following points

- It does not use optical flow or estimate incremental motion,
- it estimates the actual travel direction of the robot, i.e. the slip angle,
- it performs landmark tracking; however the FTrace system does not use natural or artificial landmarks in a conventional sense, but it looks at the traces of the vehicle, which can be considered as special landmarks, generated by the vehicle itself,
- trace tracking is performed using a Hough transform detector within a fuzzy inference model.

Previous research in the field of sinkage estimation has measured the change in an articulated suspension's configuration for the purposes of improving odometry and determining the sinkage of wheels relative to one another (Wilcox, 1994). However, absolute sinkage is necessary for mobility analysis and terrain identification algorithms. Potentially, wheel sinkage can be estimated using visual odometry methods (Olson et al., 2001). Practical implementation of these methods may be relatively complex since they rely on the use of feature identification and tracking algorithms. Brooks et al., 2006 also proposed an online visual sinkage estimation algorithm that relies on the analysis of greyscale intensity along the wheel rim. Assuming that the wheel has a different colour than the terrain, the location of the terrain interface is computed as the point of maximum change in intensity. This method is relatively simple and computationally efficient, but it is sensitive to lighting variations and shadows. Moreover, it is based on the assumption that the wheel has a different grey level than the terrain, which implies previous knowledge of the soil appearance characteristics. Conversely, the proposed method does not require any a priori information about the environment, while preserving computational efficiency.

2. Longitudinal Slip

The greatest enemy of odometric accuracy is wheel-slippage, and vehicles that travel on sandy soils are at risk the most. This is particularly true for rover-like vehicles due to their overconstrained nature, i.e., having more independent motors than degrees of freedom of motion. For these vehicles, any momentary mismatch between wheel velocities with respect to the vehicle kinematic constraints will result in wheels "fighting" each other, and, consequently, slippage with ill-effects such as position estimation errors, increased power consumption, and loss of traction.
In order to detect longitudinal Wheel Slippage (WS), a set of so-called "WS indicators" was developed in (Reina et al., 2006). The general approach is based on observing many different on-board sensor modalities and defining deterministic conditions for wheel slippage. The output of a WS indicator can be a binary flag that indicates that WS has occurred, or it can be a fuzzy quantity that expresses our certainty that WS occurred. Fuzzified outputs from multiple indicators can be combined through fuzzy logic. The most effective WS indicators are:

- *Encoder Indicator* (EI): compares encoder readings with each other.
- *Gyro Indicator* (GI): compares encoder readings with those of the *z*-axis gyro.
- *Current Indicator* (CI): monitors motor currents, which are roughly proportional to the external torque applied to each wheel.

In the remainder of this section, each indicator is discussed in some detail showing experimental results.

2.1 Encoder Indicator

Figure 2 shows a schematic diagram of a six-wheel rover. A coordinate system is attached to the vehicle so that the *y*-axis of the vehicle coordinate system is aligned with the vehicle's longitudinal direction. Each wheel *i,j* has a linear velocity vector $V_{i,j}$ and a steering angle $\varphi_{i,j}$ (index *i* = Front, Center, or Rear, and index *j* = left or right). $\varphi_{i,j}$ is measured between the longitudinal direction of the vehicle Y_b and the steering direction of the wheel. The projection of the speed vector $V_{i,j}$ onto the y-axis is called "longitudinal velocity component" $V'_{i,j}$. While the linear velocities of the wheels differ from each other according to their distance from the Instantaneous Center of Rotation (ICR) of the vehicle, their longitudinal components must be equal on either side of the vehicle.
The underlying hypothesis is that unequal longitudinal velocity components in the three

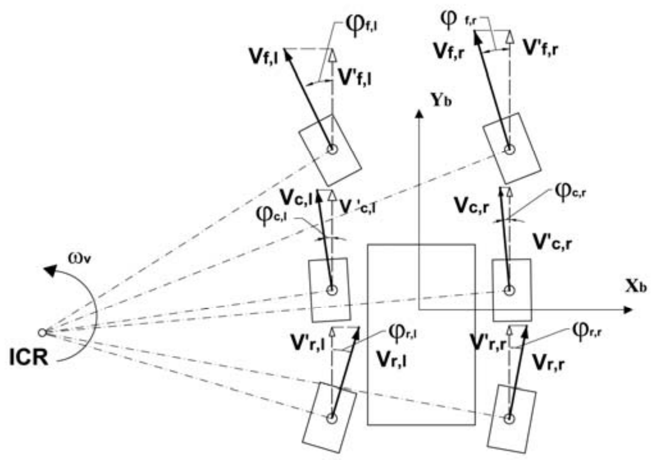


Fig. 2. Nomenclature for wheel velocities in a six-wheel rover

wheels of a side suggest wheel slippage. In order to express this hypothesis, fuzzy logic can be adopted. Details of the fuzzy inference engine can be found in (Reina et al., 2006). The fuzzy data fusion uses three inputs and one output. Inputs are the sensory data, i.e., the differences in longitudinal velocity components between the front and the center wheel, the front and the rear wheel, and the center and the rear wheel, respectively. The output is a dimensionless factor ranging from zero to one that expresses the degree of confidence we have that WS has occurred.

As an example, Fig. 3 shows the output of the fuzzy logic system of the EI for (a) a high-traction and (b) a high-slippage terrain.

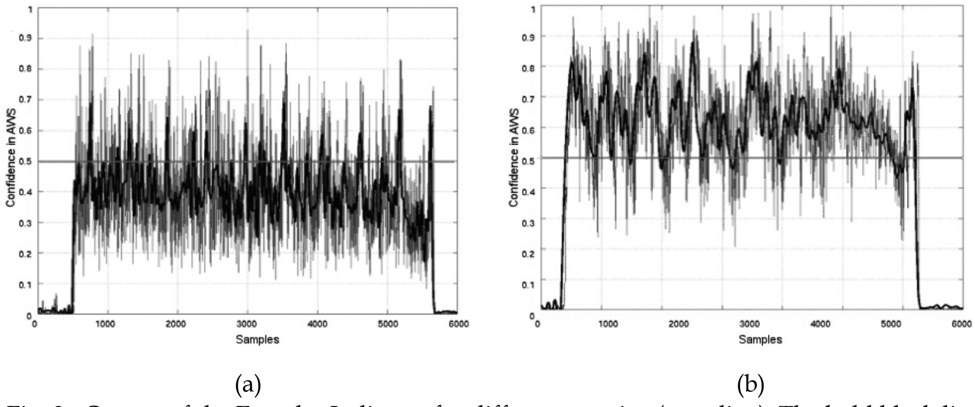


Fig. 3. Output of the Encoder Indicator for different terrains (gray line). The bold black line shows the smoothed output. a) High-traction terrain, no slippage. b) Sloped, sandy terrain causing lots of slippage

2.2 Gyro Indicator

This method aims to detect wheel slippage by comparing encoder data with gyro data. The motion of a rigid body can always be expressed as a momentary pure rotation about the ICR. For straight-line motion, the distance from the ICR to each wheel is of infinite length. The rate-of-turn ω of the vehicle can be computed from each one of the encoder pair i (index i = front, center, and rear), according to:

$$\omega_{Enc,\,i} = \frac{d_{i,r} \cdot \cos\varphi_{i,r} - d_{i,l} \cdot \cos\varphi_{i,l}}{b \cdot T} \tag{1}$$

Where:

$d_{i,\,r/l}$ - distance traveled by the right/left wheel of wheel pair i;
$\varphi_{i\,,r/l}$ - steering angle of the right/left wheel of wheel pair i;
b - vehicle width (distance between the left and right wheel) ;
T - sampling interval.

Each of the three $\omega_{Enc,i}$ can be compared with the rate-of-turn measured by the z-axis gyro ω_{Gyro}, which we consider the ground truth in this approach. If no slippage occurred in a wheel pair, then one can expect good correspondence between the rate-of-turn derived from the encoders of that wheel pair and the gyro. Poor correspondence suggests wheel slippage. Also for the Gyro Indicator, a fuzzy inference system to fuse sensory data was developed (Reina et al., 2006). Figure 4 shows the output of the fuzzy logic system of the Gyro Indicator for a test where the rover was commanded to travel with a constant rate-of-turn of 4°/s on sand. During the run, the wheels were deliberately forced to slip by manually holding back the vehicle using strings attached to its frame. The bold black line is the ground truth, provided by the gyro. The black, dotted grey, and grey lines are the rates-of-turn estimated by the front, centre, and rear encoder pairs, respectively. The output of the GI is expressed in terms of a binary state, which we refer to as an "WS flag" The WS flag is raised whenever the system's confidence in the existence of an WS condition, as estimated by the fuzzy fusion system, is greater than 0.5. This is shown by a grey line in the bottom part of Fig. 4.

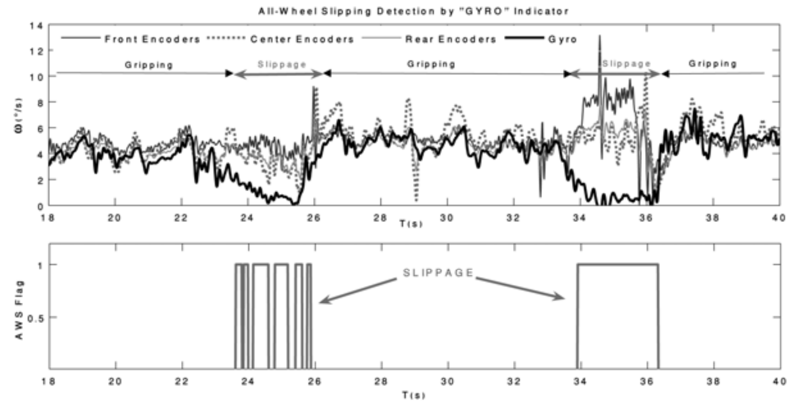


Fig. 4. Effectiveness of the Gyro Indicator

2.3 Current Indicator

The Current Indicator detects WS through the use of the well-established physical model of wheel-terrain interaction mechanics. The interaction between wheel and terrain has been shown to play a critical role in rough terrain mobility (Bekker, 1969). When a torque is applied to the wheel, shearing action is initiated at the wheel-terrain interface creating a force *F*, which is used to propel the vehicle and overcome the rolling resistance (see Fig. 5). *F* is usually referred to as the "tractive effort." The maximum tractive effort that can be developed by a wheel is determined by the shear strength of the terrain according to the Coulomb-Mohr soil failure criterion:

$$F_{Max} = A \cdot (c + \sigma \cdot \tan\varphi) = A \cdot c + W \cdot \tan\varphi \tag{2}$$

where:

c, φ	-	cohesion and internal friction angle, respectively. These coefficients characterize the behaviour of the terrain;
A	-	wheel contact patch, which is a function of wheel geometry and of the vertical load acting on the wheel;
σ	-	normal component of the stress region at the wheel-terrain interface (Fig. 5);
W	-	vertical load acting on the wheel.

Since torque is proportional to current, the knowledge of the maximum allowable shear strength of a given terrain allows estimation of the electrical current, I_{max}, that is drawn by the wheel drive motor corresponding to the maximum tractive effort. This current is so-called the "*maximum traction current*." In practice, I_{max} can be determined experimentally for a given terrain, and the condition of total slippage (100%) of the wheel is evaluated as:

$$\left|I_{\max} - I_i\right| \le \Delta I \tag{3}$$

where I_i is the current drawn by the motor of wheel *i* and ΔI is an empirically-determined threshold (in our system: 10% of I_{max}). One should note that I_{max} is a function of the vertical load acting on the wheel for a given terrain. For example, when the vehicle is traversing a slope, I_{max} will decrease since less of its weight acts as a force normal to the surface. This is accounted for by increasing ΔI (thus lowering the current threshold that suggests slippage) by an empirically determined factor C_θ. Therefore, (3) can be rewritten as:

$$\left|I_{\max} - I_i\right| \le \Delta I + C_\theta \cdot \theta_r \tag{4}$$

where θ_r is the rover's pitch angle as measured by the onboard IMU. It should be noted that (4) could only be used for predicting the maximum tractive effort for a wheel. However, methods for evaluating quantitatively the amount of slippage and thus compensating for position estimation errors have been proposed by Ojeda et al., 2006 where a relationship between the electrical currents and wheel slip was obtained to adjust corrupted encoder readings. Errors were reduced well under 1% of the total travel distance and by up to one order of magnitude when compared to conventional dead-reckoning.

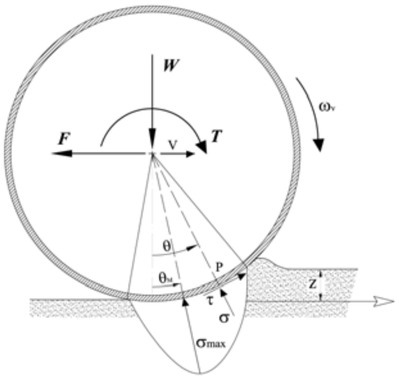


Fig. 5. Wheel-soil interaction model (adapted from (Bekker, 1969)).

2.4 Experimental Results

In this section, experimental results are presented to validate the proposed approach. The effectiveness of the WS indicators was tested on a rover operating in a laboratory sandbox. In this experiment, the robot was commanded to follow a 4-meter straight traversing two consecutive sandy mounds (as shown in Fig. 6), resulting in alternating and varying episodes of slippage. The height of the mounds was about 20 cm.

The slippage detection based on the WS indicators is expressed in terms of a binary output in the form of what we call the WS flag. When the slippage condition established by the indicator is simultaneously met by all the wheels of the vehicle, then the WS flag is raised. The WS flag is lowered when at least one wheel of the vehicle is "gripping." It should be noted that on each of the four corner of the sandbox, an ultrasonic receiver was installed. Together with a star-like formation of four ultrasonic transmitters mounted on the rover, they form a ground truth position measuring system. Within the confined area of our sandbox this system provides absolute position information in real-time and with sub-centimeter accuracy (ground truth).

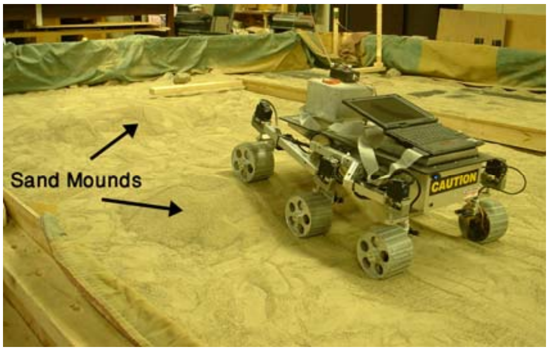


Fig. 6. The rover negotiates a sand mound along a 4-meter path at the Mobile Robotics Laboratory of the University of Michigan

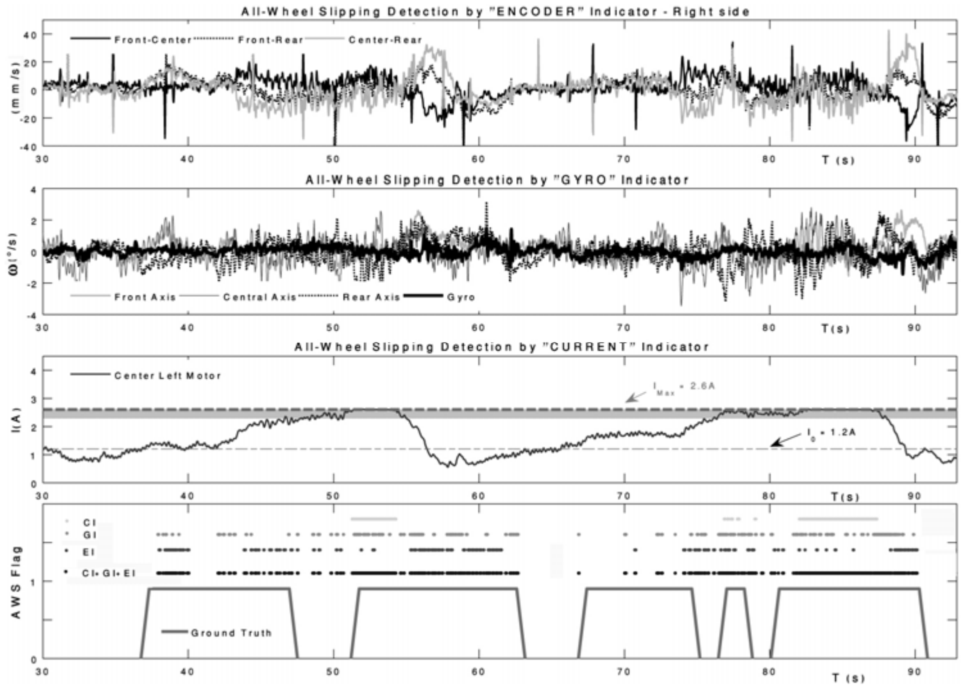


Fig. 7. Effectiveness of the EI, the GI, and the CI during the negotiation of 4-meter path with two sand mounds

The original sensor signals and the output of the indicators are plotted in Fig. 7. The upper plot shows the sensor signals used in the Encoder Indicator for the right side of the vehicle. The black, dotted black, and gray lines are the differences in longitudinal speed between, respectively, the front and the center, the front and the rear, and the center and the rear wheel. The output of the EI is the AWS flag shown by black dots in the bottom graph of Fig. 7. The second plot shows the sensor signals in the GI. Again, the bold black line is the ground truth for this indicator, provided by the gyro.
The gray, dark gray, and dotted black lines are the rates-of-turn as computed by the front, center, and rear encoder pairs, respectively. The output of the GI is the WS flag shown by dark gray dots in the bottom graph of Fig. 7. The third plot shows the electrical current measured in the center left motor. When the slippage condition expressed by (4) is met for all the wheels, then the current-based WS flag is raised. This flag is shown as the gray dots in the bottom graph. We can compare the accuracy of the EI, the GI and the CI WS flags can be compared to the ground truth flag, shown by a continuous gray line in the bottom part of Fig. 7. For this experiment, we found that the EI was correct for 25% of the time whereas the GI flagged WS correctly 38% of the time. The CI flagged WS correctly only 18% of the time. When the three flags were logically OR-ed, the indicators were correct 61% of the time.

3. Lateral Slip

A novel approach for slip angle estimation of mobile robots travelling on loose terrain was presented in (Reina et al., 2010). It uses a rearward-facing video camera to measure the pose of the trace that is produced by the wheels, and detect whether the robot follows the desired path or deviates from it because of slippage. Figure 8 shows a direct example that will help to clarify this method. Figure 8(a) shows a four-wheel rover as driving up a sandy slope following a planned straight path without any significant sideslip. This is shown by two distinct traces parallel to the direction of motion and produced by the wheel pair of either side of the robot. However, if a transverse inclination of the terrain is also present, the consequent external side force acting on rover's body will result in a substantial lateral drift, as shown in Fig. 8(b). The traces, left by the wheels of the same side of the robot, are no longer superimposed and, more importantly, their angle of inclination, with respect to a reference frame attached to the vehicle, differs from the case of absence of slip. The proposed approach aims at estimating the slip angle of the robot by measuring the pose of one of the wheel traces, i.e. the rear left wheel, in conjunction with the knowledge of the rate-of-turn provided by an onboard sensor.

3.1 The FTrace system

The presence of shadows, occlusions, and terrain unevenness can turn trace tracking into a very complex problem. A robust and efficient trace detection system must be able to filter out all disturbances and extract the marking of interest from a non-uniform background. In Fig. 9, a sample image set demonstrates the variety of terrain and environmental conditions that can be encountered. Fig. 9 (a) shows a scene where trace detection can be considered relatively easy thanks to the clearly defined imprint and uniform terrain texture. In Fig. 9 (b), extraction of wheel trace is more difficult due to the presence of terrain discontinuities, whereas Fig. 9 (c) shows a non-uniform terrain texture. A more complex wheel trace is shown in Fig. 9 (d) and Fig. 9 (e), due to the presence of heavy shadowing. Finally, Fig. 9 (f) refers to a condition where two imprints are present in the same scene.

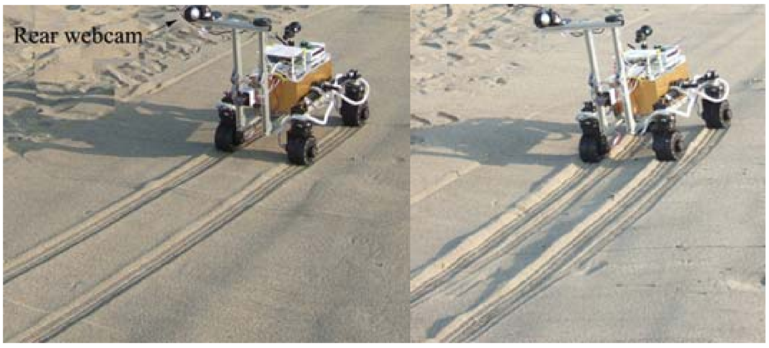


Fig. 8. Sideslip estimation for the robot El-Dorado, built at the Space robotics Lab of Tohoku University, by visual observation of the wheel traces with a rear webcam: (a) wheel traces parallel to the direction of motion, no lateral slip, (b) wheel traces inclined with respect to the intended direction of motion, significant lateral slip.

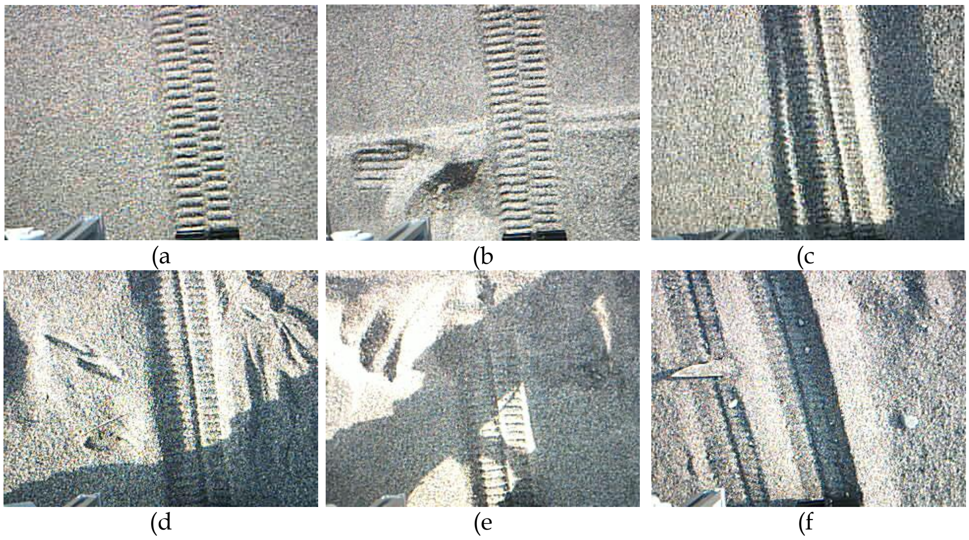


Fig. 9. Sample images of terrain and wheel trace conditions: (a) uniform sandy terrain, (b) disturbances due to transversal line-like discontinuities, (c) non-uniform terrain texture, (d) and (e) non-uniform terrain texture with noise due to shadowing, (f) different trails present in the scene.

The FTrace module performs two main tasks. For more details, the reader is referred to (Reina & Yoshida, 2008):

- Extraction of trace candidates from the camera-acquired image and estimation of their pose with respect to the sensor, i.e. the vehicle, reference frame.
- Selection of the candidate that best fits to the trace model.

In order to determine which line best fits to the trace model, the system employs a robust Hough transform enhanced by fuzzy reasoning. The general approach is based on comparing the geometrical properties of each candidate with those of the trace model in both the image plane of the camera and the real world, and defining deterministic conditions for model matching. The output of the FTrace is a fuzzy quantity that expresses the certainty that the line agrees exactly with the trace model. As a representative result, Figure 10 shows a sample image referring to a field trial on sandy terrain. The FTrace system was able to extract ten candidates from the original scene (Fig. 10(a)), as shown in Fig. 10(b) and Fig. 10(c). The trace selection stage provided the confidence matches collected in Table 1. The lane marker, denoted with L_5 in Fig. 10(b) and shown by a dotted line in Fig. 10(c), yielded the greatest confidence level (89.0%), and was therefore selected as the best match. In Fig. 10(d), the output of the FTrace system is overlaid over the original image along with the estimated values of lateral slip.

3.1 Field Validation

In this section experimental results are presented, aimed at validating our approach for lateral slip estimation by visual observation of the wheel trace.

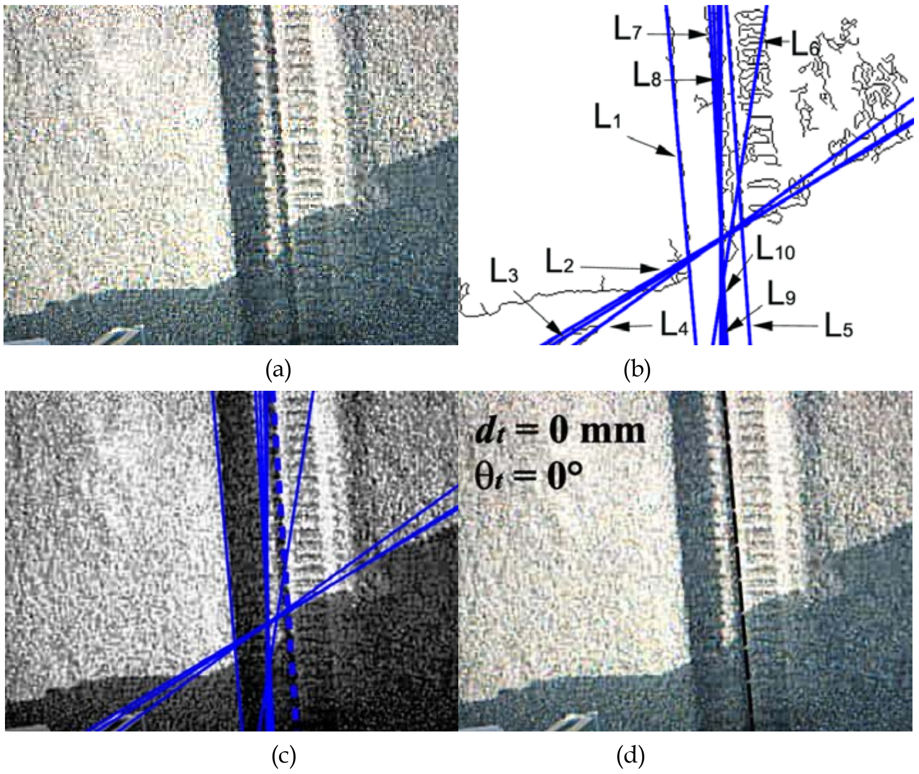


Fig. 10. Application of the FTrace system to a sample image: a) original scene, b) and c) application of edge detection and Hough transform, d) output of the FTrace system. Note that for this image no sideslip was detected.

The FTrace system was tested in the field using a four-wheel rover, outfitted with a cost-effective webcam acquiring at 5 Hz and with a field of view on the ground corresponding to a 60 cm long × 80 cm wide area, behind the left rear wheel. The test field was located on the shoreline of a sandy beach, comprising large flat areas and sparse mounds of different extensions and heights. In all experiments, the rover was remotely controlled using a wireless joypad, and commanded to follow a straight-line path with a travel speed of about 8 cm/s. Two types of environment were considered:

- Set A: sandy relatively flat terrain. These experiments were aimed at evaluating undue errors of the FTrace system incurred by low-slippage terrain.
- Set B: sandy non-flat terrain, including driving uphill or sideways on sandy slopes with substantial lateral slip.

The entire experimental area was within the distance range (order of tens of meters) of a laser-based absolute positioning system that provided the ground-truth.
The FTrace system was tested over a total of 15,016 images (about 250 m of overall travel distance) showing the results collected in Table 2 for both sets of experiments. The percentage of false positives, i.e. a trace marker detected when actually there is no trace

marker, was less than 0.3%, and limited to the initial moments of robot's motion, when no wheel track was present on the ground yet. Note that in the initial stage, the FTrace system relies only on Hough transform to extract strong lines from the scene, which pass through the wheel centre, whereas the fuzzy inference system kicks in after the rover begins travelling. Conversely, false negatives arise when the trace marker is present in the image but the system is not able to detect it at all and does not return any information.

SET#	Frames	False Positives (%)	False Negatives (%)	Misid. (%)
A	2560	0.1	1.5	0.0
B	12456	0.2	2.7	0.8

Table 1. Results obtained from the FTrace system for different types of terrain. Set A: flat sandy terrain. Set B: non-flat sandy terrain

Candidate Line #	L_1	L_2	L_3	L_4	L_5	L_6	L_7	L_8	L_9	L_{10}
Confidence Match (%)	4.1	1.1	1.3	1.4	89.0	9.1	20.0	20.3	18.7	20.2

Table 2. Degree of confidence in model matching for the wheel trace candidates of Fig. 10, as derived from the FTrace system.

The percentage of false negatives was less than 3%, and due largely to poor image segmentation and camera calibration errors (80%), and model approximation (20%). In those cases the last detected wheel trace is retained for comparison with the successive scenes. This approach proved successfully since the number of consecutive false negatives was always less than 3 (< 0.6 s). Finally, misidentifications refer to cases in which a trace marker is present in the image, but the system fails in recognizing it properly and returns wrong information. In all tests, misidentifications were less than 1%. Overall, the system proved to be robust to disturbances due to heavy shadowing, non-uniform terrain texture, and the presence of overlapping imprints.

The accuracy of the FTrace system in estimating the lateral drift of the robot was assessed in an experiment on non-flat terrain. The rover was driven up an approximately 5-degree sandy slope. During its path, the robot climbed sideways on a large mound, resulting in continuous and almost constant lateral drift. Figure 11 shows the position of the robot and the imprints produced by its wheels at the end of the run, from a front and rear view, respectively. The total travel distance resulted in about D=7 m. The slip angle, derived from the FTrace system, is compared with the ground-truth data in Fig. 12. The two curves show good agreement with a root mean square (RMS) error less than 1.5 deg.

The FTrace system detected effectively the onset of sideslip and its successive trend throughout the experiment. Two different stages can be distinguished during the test. In the first stage, referring to the first 60 seconds, the robot followed its intended straight path without any significant drift. This is demonstrated by the two wheel traces parallel to the direction of motion. The second stage marks the onset of sideslip caused by the external lateral force acting on the rover due to the transverse inclination of the terrain. As a direct consequence, the angle of inclination of the wheel traces changes, (see also Fig. 11(b)), attesting to the feasibility of this approach.

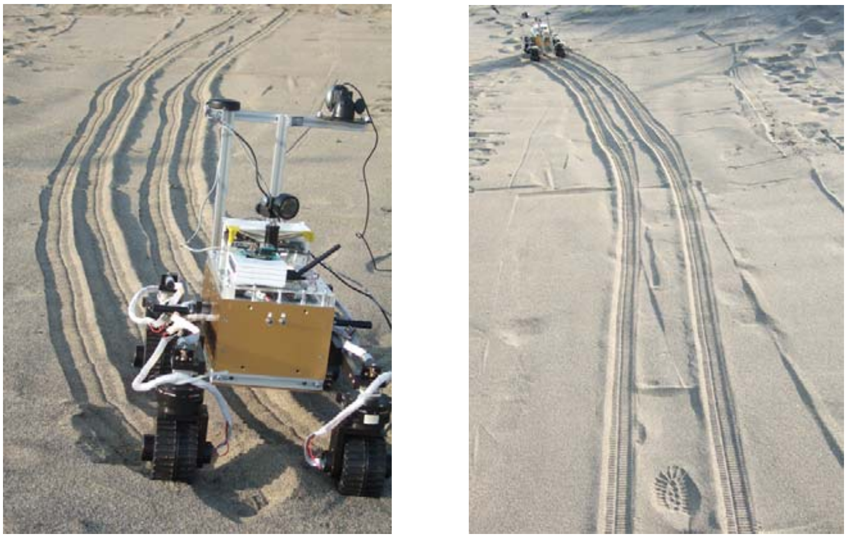


Fig. 11. Position of the rover and wheel traces at the end of the traverse of a sandy slope: a) Front view, b) rear view

In order to allow useful dead-reckoning even in presence of lateral drift, a method for slippage compensated odometry was presented by Reina et al., 2010 using an integrated longitudinal and lateral friction model. This approach was proved to be effective in correcting odometry errors caused by wheel slippage, during traverse of sandy slopes.
As a representative result, Fig. 13 shows the path followed by the robot as estimated from plain odometry by a dash black line, for the previously-described test. The path derived from the slippage-compensated odometry is shown by a solid gray line. Both data can be compared with the ground-truth shown by a solid black line. An absolute radial error *E* was computed for the estimation of the final position of the robot. The error in the estimation of the final position of the robot using odometry was E_e=2.84 m. If odometry is enhanced by measuring the actual heading of the robot with an onboard compass rather than wheel encoders, the error was decreased to E_{oc}=2.50 m. When using the sComp odometry the final position error was limited to a remarkable E_{sC}=0.14 m with an overall reduction of 95% with respect to plain odometry, and 83% compared to compass-enhanced odometry.

4. Wheel Sinkage

The algorithm aims to measure wheel sinkage in deformable terrain from an image of the wheel. It is assumed that a camera is attached to the wheel assembly with a field of view containing the wheel-terrain interface, and that its location relative to the wheel is known and fixed during vehicle travel. The wheel under inspection needs to be outfitted with a pattern composed of equally spaced concentric circumferences on a white background, as shown by the practical implementation of the visual sinkage estimation (VSE) system in Fig. 14.(a). Since the wheels of off-road robots are mostly customized, this should not require much effort. Sinkage *z* can be estimated by measuring the contact angle between wheel and terrain. The algorithm relies on a relatively simple edge-detection approach. The idea is to identify the wheel radial lines, along which the number of edges is less than that expected when the wheel rolls without sinkage. Those lines can be associated with the part of the wheel obscured by terrain and thus with sinkage, as shown in the explanatory scheme of Fig. 14. (b).

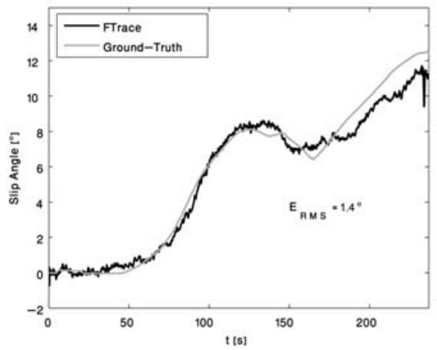


Fig. 12. Effectiveness of the FTrace system in estimating the robot slip angle during sideways traverse of a sandy slope

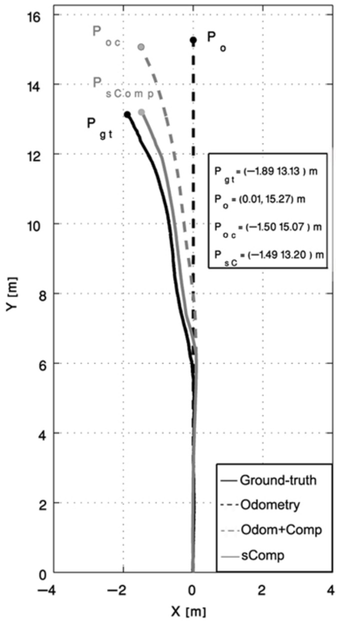


Fig. 13. Position estimation of the rover path during the traverse of a sandy slope

One should note that alternative configurations of this approach may be thought and implemented, using, for example, a belly-mounted camera. This would fix the potential drawback of having an off-side-sensor. To determine the contact angle, only the annular region along the wheel rim and including the pattern needs to be examined. By limiting the analysis only to this region of interest, the accuracy of the algorithm can be improved and its computational burden reduced. More details of the visual algorithm can be found in (Reina et al., 2006).

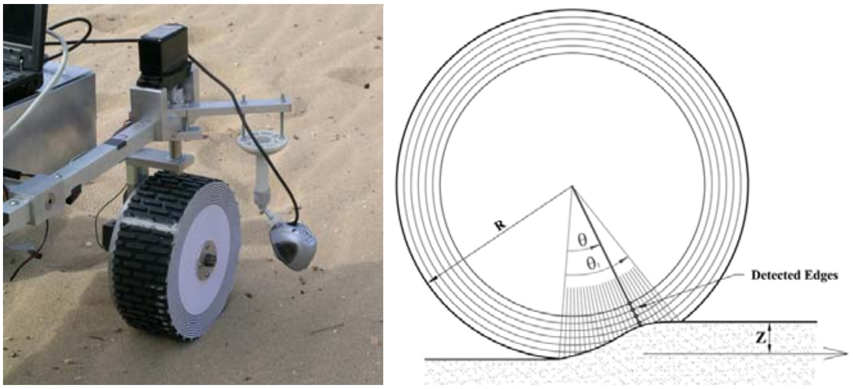
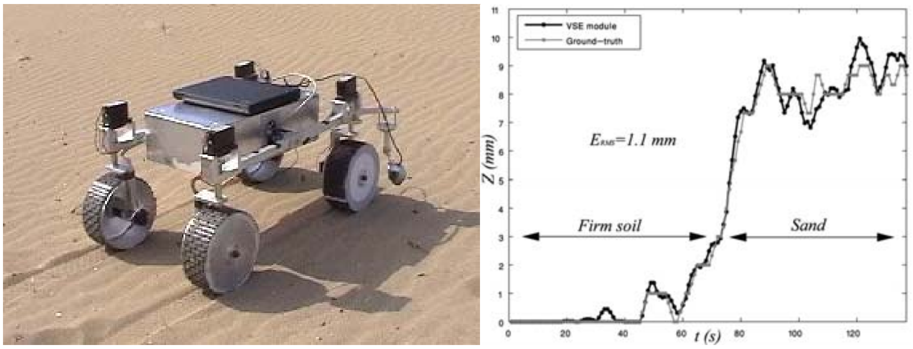


Fig. 14. Robot wheel outfitted for the VSE system (a), strategy for sinkage estimation (b)

The VSE system was validated using a single-wheel test bed. Sinkage was successfully estimated over a total of 31,000 images with a root mean square error of 2.0 mm. the percentage of false positives, that is, sinkage detected when actually there is no wheel sinkage, was less than 0.3%, mostly due to rocks or terrain unevenness occluding the wheel-terrain interface. Conversely, false negatives arise when the wheel experiences sinkage but the system is not able to detect it at all and does not return any information. No false negatives were detected in all the experiments. The algorithm also proved to be very robust against variations of lighting conditions up to lighting reduction as much as 90% of the optimal value.



(a) (b)

Fig. 15. Sinkage estimation in the field: (a) VSE system integrated with the University of Salento rover Dune, (b) sinkage measurement

The visual algorithm was also tested in the field using an all-terrain rover operating in natural outdoor terrains. The test field was located on the shoreline of a sandy beach (as shown in Fig. 15.(a)), comprising a few areas of firm, rocky terrain. A set of experiments was performed to prove the ability of the rover in detecting changes in the physical properties of the traversed terrain, based on the measure of wheel sinkage. Under the assumption of almost constant vertical load acting on the wheel, sinkage is expected to increase in soft soils than in firm terrains. This on-board capability would enable the rover to adopt proper actions to revise its motion plan and avoid hazardous highly-deformable terrains, or modulate wheel torque to improve traction and limit slippage.

In all experiments, actual values for the wheel sinkage were manually measured by a human analyst, based on the acquired images. Figure 15(b) shows the sinkage as estimated by the VSE module in a representative test, where the rover started its path from firm terrain, and then moved on dry sand with a constant travel speed of 10 cm/s. As expected, sinkage was null during the robot motion on rigid surface, and increased as the vehicle travelled on sandy soil. The vision-based measurement matches very well the ground-truth with a RMS error of 1.1 mm.

5. Conclusions

This chapter discussed methods for wheel slip and sinkage estimation to improve mobility of rough terrain vehicles and planetary rovers. Specifically, novel measures for longitudinal slippage identification were introduced, which compare readings from encoders with each other, with data from a gyroscope, and with current sensors mounted on onboard the vehicle. It was shown that this method was effective in experimental trials on sandy, non-flat terrain. An innovative method for sideslip estimation was presented, based on observing the wheel trail left by a robot during its traverse of loose sandy terrains. Comprehensive experiments in the field demonstrated the overall effectiveness of the proposed approach. The FTrace method was able to measure the slip angle of the robot with a worst-case of less than 3% of failed observations and 1.6 deg of accuracy. Finally, a vision-based algorithm for wheel sinkage estimation was also proposed and shown to be computationally efficient, relatively accurate with maximum errors below 15%, and very robust to disturbances and variations in lighting condition.

These techniques can be used to gain important information about the vehicle-terrain interaction and to improve dead-reckoning accuracy and traction control in rough-terrain autonomous vehicles.

6. References

Baumgartner, E.T., Aghazarian, H., & Trebi-Ollennu, A., (2001). Rover Localization Results for the FIDO Rover, *Proceedings of the Conf. Sensor Fusion and Decentralized Control in Autonomous Robotic Systems*, Vol. 4571, pp. 34-44, Newton, MA, USA.

Bekker, G., (1969). *Introduction to Terrain-Vehicle Systems*, Ann Arbor, University of Michigan Press.

Brooks, C.A., Iagnemma K. & Dubowsky, S., (2006). Visual Wheel Sinkage Measurement for Planetary Rover Mobility Characterization, *Autonomous Robot*, Vol. 21, pp. 55-64.

Fuke, Y., & Krotkov, E., (1996). Dead reckoning for a lunar rover on uneven terrain, *Proceedings of the IEEE Int. Conf. on Robotics and Automation*, Vol. 1, pp. 411-416.

Gustafsson, F., (1997). Slip-based tire-road friction estimation. *Automatica*, Vol. 33(6), pp. 1087-1099.

Helmick, D., et al., (2006). Slip Compensated Path Following for Planetary Exploration Rovers. *Advanced Robotics*, Vol. 20(11), pp. 1257–1280.

Lindemann, R.A. & Voorhees, C.J. (2005). Mars Exploration Rover Mobility Assembly Design, Test, and Performance, Proceedings of the IEEE Conf. Systems, Man, Cybern., Waikoloa, Hi, pp. 450-455.

Iagnemma, K. & Dubowsky, S., (2004). *Mobile Robots in Rough Terrain: Estimation, Motion Planning, and Control with application to Planetary Rovers*. Springer Tracts in Advanced Robotics (STAR) Series, Vol. 12, Springer.

Konolige, K., Agrawal, M., & Sola, J., (2007). Large-Scale Visual Odometry for Rough Terrain, *Proceedings Int. Symposium on Research in Robotics*, Hiroshima , Japan.

Maimone, M., Cheng, Y., and Matthies, L., (2007). Two Years of Visual Odometry on the Mars Exploration rovers, *Journal of Field Robotics*, Vol. 24(3), pp. 169-186.

Olson, C., Matthies, L., Schoppers, M., & Maimone, M. (2001). Stereo ego-motion improvements for robust rover navigation, *Proceedings of the IEEE International Conference on Robotics and Automation*.

Ojeda, L., Reina, G. & Borenstein, J. (2004). Experimental Results from FLEXnav: An Expert Rule-based Dead-reckoning System for Mars Rovers, *Proceedings of IEEE Aerospace Conf.*, Big Sky, MT, USA.

Ojeda, L., Cruz, D., Reina, G. & Borenstein, J. (2006). Current-based slippage detection and odometry correction for mobile robots and planetary rovers. *IEEE Transactions on Robotics*, 22(2), pp. 366-378.

Reina, G., Ojeda, L., Milella, A. & Borenstein, J. (2006). Wheel Slippage and Sinkage Detection for Planetary Rovers. *IEEE/ASME Transactions on Mechatronics*, Vol. 1(2), pp. 185-195.

Reina, G. & Yoshida, K., (2008), Slip Angle Estimation for Lunar and Planetary Robots. *Int. Journal of Intelligence Control and Systems*, Special Issue on Field Robotics, 13(1), pp. 15-24 .

Reina, G., Ishigami, G., Nagatani, K. and Yoshida, K., (2009). Odometry Correction Using Visual Slip-Angle Estimation for Planetary Exploration Rovers, accepted for publication in *Advanced Robotics*, 2010.

Robert Bosch GmbH (2000). *Automotive Handbook*, 5th ed., Germany.

Wilcox B., (1994). Non-geometric hazard detection for a mars microrover, *Proceedings of the AIAA Conf. Intelligent Robotics in Field, Factory, Service, and Space*, Vol. 2, pp. 675-684. Houston, TX, USA.