A COLLABORATIVE MULTI-ROBOT LOCALIZATION TECHNIQUE FOR AUTONOMOUS ROBOTS

by

Hatice Köse Bağcı

B.S. in Computer Engineering, Boğaziçi University, 1997

M.S. in Computer Engineering, Boğaziçi University, 2000

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Computer Engineering

Boğaziçi University

2007

A COLLABORATIVE MULTI-ROBOT LOCALIZATION TECHNIQUE FOR AUTONOMOUS ROBOTS

APPROVED BY:

Prof. H. Levent Akın (Thesis Supervisor)	.....
Prof. H. Işıl Bozma	.....
Prof. Okyay Kaynak	.....
Prof. A. C. Cem Say	.....
Prof. A. Coşkun Sönmez	.....

DATE OF APPROVAL: 29/12/2006

ACKNOWLEDGEMENTS

I would like to specially thank to my thesis supervisor, Prof. H. Levent Akın, for his endless motivation, patient guidance, and understanding. This thesis would not be possible without his contributions.

Many thanks to my sincere jury members Prof. H. Işıl Bozma, and Prof. A. C. Cem Say for their kind and patient contributions to my thesis. In addition, I would like to thank to Prof. Okyay Kaynak and Prof. A. Coşkun Sönmez, for kindly attending to my thesis defense.

Special thanks to Gal Kaminka, Dieter Fox, Stephen Gutmann, and Olivier Mitchell, and for the others who kindly helped when necessary. In addition, thanks to Kerstin Dautenhahn, Chrystopher Nehaniv, and STRI members for their kind and patient motivation.

Many thanks to all my instructors in CMPE department, for their endless motivational and scientific support, and thanks to the residents of AILab and Cerberus Team, who supported me with their friendship and professional collaboration.

This work was supported in part by Boğaziçi University Research Fund projects 01A101 and 03A101D, and State Planning Agency project DPT 03K120250.

Special thanks to my parents, Gül and Mehmet Köse, for their motivation, kindness and help during this study, and for the rest of my life. Their motivation and support made this thesis and my dreams true.

This thesis is dedicated to my dear daughter Ayşe Alanur Bağcı, who deserves the most, by letting her mother work, in their precious play time. Lastly, special thanks to my husband Sadrettin Bağcı, for being in my life.

ABSTRACT

A COLLABORATIVE MULTI-ROBOT LOCALIZATION TECHNIQUE FOR AUTONOMOUS ROBOTS

This work proposes a novel method for collaborative global localization of a team of soccer playing autonomous robots. It is also applicable to other indoor real-time robot applications in noisy, unpredictable environments, with insufficient perception.

A novel solution, *Reverse Monte Carlo Localization (R-MCL)* is designed to solve single self-localization problem using local perception and action about the surrounding environment for each robot. R-MCL is a hybrid method based on *Markov Localization (ML)* and *Monte Carlo Localization (MCL)* where the ML based part finds the region where the robot should be and the MCL based part predicts the geometrical location with high precision by selecting samples in this region.

In the multi-robot localization problem, robots use their own local position estimations, and the shared information from other team mates, to localize themselves. To integrate the local information and beliefs optimally, avoid conflicts and support collaboration among team members, a novel collaborative multi-robot localization method called *Collaborative Reverse Monte Carlo Localization (CR-MCL)*, based on R-MCL, is presented. When robots detect each other, they share the grid cells representing this observation. The power of the method comes from its hybrid nature. It uses a grid based approach to handle detections which can not be accurate in real-time applications, and sample based approach in self-localization to improve its success, although it uses lower amount of samples compared to similar methods. Both methods are tested using simulated robots and real robots and results show that they are fast, robust, accurate and cheap in terms of communication, memory and computational costs.

ÖZET

OTONOM ROBOTLAR İÇİN BİR ÇOKLU ROBOT KONUŞLANDIRMA TEKNİĞİ

Bu çalışma bir otonom robotlar takımının dayanışmalı konuşlandırılması için yeni bir yöntem önerir. Bu çalışma aynı zamanda gürültülü, önceden tahmin edilemez ve yeterli algılama yapılamayan diğer iç mekan gerçek-zamanlı robot çalışmaları için de uygundur.

Tekli kendini konuşlandırma sorununu her robot için bulunduğu çevreye ait yerel algı ve hareket bilgisi kullanılarak çözmek için Ters Monte Carlo konuşlandırması(R-MCL) isimli yeni bir yöntem tasarlanmıştır. R-MCL, Markov konuşlandırması(ML) ve Monte Carlo konuşlandırması(MCL) üzerine kurulan melez bir yöntemdir. Bu yöntemde ML tabanlı bölüm robotun olması gereken bölgeyi bulur ve MCL tabanlı bölüm bu bölgeden örnekler seçerek geometrik konumu yüksek çözünürlükle bulur.

Çoklu robot konuşlandırması sorununda robotlar kendi yerel tahminlerini ve diğer takım arkadaşlarından gelen paylaşılan bilgiyi kullanarak kendilerini konuşlandırır. Yerel bilgi ve inançları uygun olarak birleştirebilmek, çatışmaları önlemek ve takım elemanları arasında dayanışmayı desteklemek için R-MCL'e dayalı yeni bir dayanışmalı çoklu robot konuşlandırma metodu sunulmuştur. Robotlar birbirlerini algıladıklarında bu gözlemi yansıtan ızgara hücrelerini paylaşırlar. Yöntemin gerçek gücü melezliğinden gelir. Gerçek-zamanlı uygulamalarda kesin olarak ölçülemeyecek gözlemleri ızgara tabanlı yaklaşımla, kendini konuşlandırma sorununu ise daha kesin sonuç veren örnek tabanlı yaklaşımla, benzer çalışmalardan daha az örnek kullanarak çözer.Her iki yöntem de benzetim robotları ve gerçek robotlar ile denenmiş ve sonuçlar yöntemlerin hızlı, sağlam, kesin ve hesap, hafıza ve iletişim masrafları açısıdan ucuz olduğunu göstermiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

a	The odometry data coming from actions
b	The bias in Fuzzy localization
$Bel(l)$	The belief that the robot is at the location l
Bel_{nm}	The belief of the robot n using the shared information coming from the robot m
G	The grid list
G‘	The new grid list
G_i	The i^{th} grid cell
Gt	The fuzzy grid map in Fuzzy localization
$Gt(x, y)$	The degree of possibility of being in (x,y) in Fuzzy localization
h	The height in Fuzzy localization
H_{xk}	The expected measurement in Kalman Filter
K	The Kalman Gain in Kalman Filter
l	The current estimated location of the robot
l‘	The estimated location of the robot before the action is executed
L	The three dimensional random variable composed of the robot's position and heading
m	The map of the environment in which the robot acts
s_n	The sensory inputs
$St(x, y\|r)$	The degree of possibility of being in (x,y) given the range r in Fuzzy localization
$p(l)$	The probability of the robot being in location l
P	The covariance in Kalman Filter
PE_x	The x-coordinate of the pose estimate before a perception or odometry update in MCL part of Reverse Monte Carlo Localization
PE_x^*	The x-coordinate of the updated pose estimate in MCL part of Reverse Monte Carlo Localization

PE_y	The y-coordinate of the pose estimate before a perception or odometry update in MCL part of Reverse Monte Carlo Localization
PE_y^*	The y-coordinate of the updated pose estimate in MCL part of Reverse Monte Carlo Localization
PE_θ	The orientation of the pose estimate before a perception or odometry update in MCL part of Reverse Monte Carlo Localization
PE_θ^*	The orientation of the updated pose estimate in MCL part of Reverse Monte Carlo Localization
$Pose_x$	The x-coordinate of the pose estimate before a perception or odometry update in Markov Localization part of Reverse Monte Carlo Localization
$Pose_y$	The y-coordinate of the pose estimate before a perception or odometry update in Markov Localization part of Reverse Monte Carlo Localization
$Pose_\theta$	The orientation of the pose estimate before a perception or odometry update in Markov Localization part of Reverse Monte Carlo Localization
r	The range
r_m	The detection information of the robot m in Collaborative Reverse Monte Carlo Localization
S'	The sample set
o	The perception data coming from observations
o_n	The perception data of robot n coming from observations
th_{ML}	The threshold of grid cells in Markov Localization part of the Reverse Monte Carlo Localization
u	The control vector in Kalman Filter
w_i	The weight for the i^{th} grid cell in Reverse Monte Carlo Localization
w	The importance weight in Monte Carlo Localization
x	The position of the robot in the x-coordination in Reverse Monte Carlo Localization
x'	The position of the robot in the x-coordination in Geometrical Localization

X	The mean in Kalman Filter
X_L	The position of the landmark in the x-coordination in Geometrical Localization
X_O	The old position of the robot in the x-coordination in Geometrical Localization
y	The position of the robot in the y-coordination in Reverse Monte Carlo Localization
y'	The position of the agent in the y-coordination in Geometrical Localization
Y_L	The position of the landmark in the y-coordination in Geometrical Localization
Y_O	The old position of the agent in the y-coordination in Geometrical Localization
Δx	The distance the robot moved in sideways in the odometry update of Reverse Monte Carlo Localization
Δy	The distance the robot moved on the line of its heading in the odometry update of Reverse Monte Carlo Localization
z	The actual movement in Kalman Filter
$2-D$	Two dimensional
$3-D$	Three dimensional
α	The slope in Fuzzy localization
α	The normalizer in Markov Localization
θ	The heading of the robot in Reverse Monte Carlo Localization
Δ	The width of the core in Fuzzy localization
$\Delta\theta$	The angle that the robot has turned in odometry update of Reverse Monte Carlo Localization

AI	Artificial Intelligence
A-MCL	Adaptive Monte Carlo Localization
CMCL	Collaborative MCL
CMRAL	Collaborative Multi Robot Active Localization
Comm.	Communication
CP	Cooperative Positioning

CPCBL	Collaborative Probabilistic Constraint-based landmark localization
CPS	Cooperative Positioning System
CR-MCL	Collaborative Reverse Monte Carlo Localization
DBN	Dynamic Bayesian Network
DMRL	Distributed Multi Robot Localization
EGO	Ego-Centric Approach
EKF	Extended Kalman Filter
GEO	Geometrical Localization
H-POMDP	Hierarchical Partially Observable Markov Decision Process
KLD	Kullback-Leibler Distance
KLD-S	Kullback-Leibler Distance-Sampling
MCL	Monte Carlo Localization
ME	My Environment
MHL	Multi Hypothesis Localization
Mix-MCL	Mixture Monte Carlo Localization
ML	Markov Localization
ML-EKF	Markov Localization - Extended Kalman Filter
POMDP	Partially Observable Markov Decision Process
R-MCL	Reverse Monte Carlo Localization
SLAM	Simultaneous Localization And Mapping
S-LOC	Simple Localization
SRL	Sensor Resetting Localization
SRL*	Modified Sensor Resetting Localization version which was used in the tests

1. INTRODUCTION

The global localization problem is the estimation of the position of a robot relative to its environment, using sensor readings and its actions. In a challenging real-time test bed like robot soccer with four-legged robots, where the sensors and the environment have uncertainties, localization results are typically erroneous and inaccurate.

In robot soccer, a robot is typically expected to find its own location using distinguishable artificial landmarks in the field, and then use this information to find the location of the interesting objects such as the ball and the goals. For such a real-time application with robots limited by on board computational resources, speedy solutions with less memory and computational resources are especially demanded. Consequently, localization is a difficult and vital problem for robot soccer.

Triangulation is the simplest localization method which uses geometry to compute a single point that is as close as possible to the actual location. However, in real world applications, a robot can never calculate where it is exactly because of the uncertainty in its sensors, and the environment. Consequently, several different approaches which estimate the position of robot probabilistically were developed to consider this uncertainty in the solutions.

The Kalman filter (Kalman-Bucy filter) is a well-known approach for this problem. This filter integrates uncertainty into computations by making the assumption of Gaussian distributions to represent all probability densities including positions, odometric and other sensory measurements. Since only one pose hypothesis can be represented, the method is unable to make global localization, and cannot recover from total localization failures (Stroupe and Balch, 2002, Stroupe *et al*, 2003, Gutmann and Fox, 2002).

Many works consider Markov localization (ML) (Burgard *et al*, 1996, Fox *et al*, 1999b, Thrun *et al*, 2005, Gutmann and Fox, 1998) which is similar to the Kalman

filter approach, but it does not make a Gaussian distribution assumption and allows any kind of distribution to be used. Although this feature makes this approach flexible, it adds a computational overhead.

The Monte Carlo Localization (MCL) is a version of Markov localization that relies on sample-based representation and the sampling/importance re-sampling algorithm for belief propagation (Thrun *et al*, 2001, Schulz and Burgard, 2001). Odometric and sensory updates are similar to ML. Most of the MCL based works suffer from the kidnapping problem, since this approach fails when the current estimate does not fit observations. There are several extensions to MCL that solve this problem by adding random samples at each iteration. Some of these methods are Sensor Resetting Localization (SRL) (Lenser and Veloso, 2000), Mixture MCL (Mix-MCL) (Gutmann and Fox, 2002), Kullback-Leibler Distance (KLD)-Sampling (Fox, 2003) and Adaptive MCL (A-MCL) (Gutmann and Fox, 2002).

The Markov Localization-Extended Kalman Filter (ML-EKF) method is a hybrid method that aims to make use of the advantages of both methods, taking into consideration the fact that ML is more robust and EKF is more accurate (Gutmann and Fox, 2002).

The Multi Hypothesis Localization (MHL) method discussed in (Kristensen and Jensfelt, 2003) aims to avoid problems caused by using a single Gaussian, by considering a mixture of Gaussians, thus enabling the representation of any given probability distribution of the robot pose.

Although there have been only a few fuzzy logic based approaches, they appear to be promising (Buschka *et al*, 2000, Köse *et al*, 2003, Köse *et al*, 2005). In these approaches, the uncertainty in sensor readings (distance and heading to beacons) is represented by fuzzy sets.

The Simple Localization (S-LOC) method is a new technique which represents every perception by a sample. The old position estimation is also represented by a

sample. This sample set together with a history based module, is used to estimate the current position of the robot (Çelik, 2005).

This work is a part of a project which aims to localize legged robots in the soccer field globally, while solving the above mentioned problems. There are several limitations and assumptions related to the rules of the Four Legged League of Robocup that make this localization problem challenging (Robocup, 2005).

The ML and MCL methods are the most widely used global localization methods in the robotic soccer domain. Both have advantages and disadvantages. In this work we first introduce *Reverse Monte Carlo Localization* (R-MCL) (Köse *et al*, 2004, Köse and Akın, 2005, Köse *et al*, 2005, Köse *et al*, 2006, Köse and Akın, 2007) for single robot localization which is a hybrid approach that aims to combine ML and MCL methods, to make use of the advantages of both, and overcome the disadvantages. The idea behind this algorithm is to converge to a part of the environment by using a coarse 2-D grid based ML algorithm and in this local area, use MCL algorithm to find the current position estimation of the robot in a fast, robust and accurate manner. This algorithm is especially successful in the kidnapping problem.

Global collaborative localization of autonomous mobile robots is a highly challenging task, which is generally more successful than single robot localization, despite its high complexity and associated communication problems. In (Stroupe and Balch, 2002) Kalman filters are used to integrate the information coming from different robots. (Roumeliotis and Bekey, 2000, Roumeliotis and Bekey, 2000, Roumeliotis and Rekleitis, 2004) divide a central Kalman filter into m Kalman filters, one for each robot, to localize all the robots in the team. In (Fox *et al*, 2000), robots share a density tree based on the samples representing the observation of the detected robot, and Monte Carlo Localization (MCL) is used to localize the robots. In (Kurazume and Hirose, 2000), some of the robots stand still to serve as landmarks for others, and observe the moving ones for localization. Fuzzy membership functions are used to estimate the position, and the shared information is integrated by means of coalition in (Canovas *et al*, 2004).

In these works it is assumed that the robots can detect and identify each other, and the observations are either accurate or can be estimated using some known distributions. There are also no false positives. Additionally, there are no conflicts between the robots that share information. In real robot applications, like robot soccer, however, most of these assumptions are not valid. All the robots are dynamic, and they can not detect and observe each other very accurately. In this work for simplicity we also assume that they can identify each other. Conflicts can arise between robots when one or more robot fails in sensing, or localizing itself, or is kidnapped. However, collaborative localization is expected to be more successful than single robot localization since it avoids single point failure, and is helpful in case of high noise and sparsity, where the robots can fine tune their estimations using the shared information from their team mates.

In this work, we also propose a novel collaborative method that aims to globally localize a team of robots, based on our single self-localization method R-MCL. Whenever two or more robots detect each other, they represent these detections in terms of grid cells, and mutually send them to each other. Production, fusion and integration of the shared data are performed by *Collaborative Reverse Monte Carlo Localization (CR-MCL)* which is a modified version of R-MCL for multi robot localization. The real power of the method comes from its hybrid nature. It uses a grid based approach to handle detections which can not be accurate in real time applications, and sample based approach in self-localization which improves its success, although it uses lower amount of samples compared to similar sample based methods.

In Chapter 2, the single-robot and multi-robot localization problems are discussed and some well-known techniques are described briefly. The R-MCL method, which is the novel solution proposed in this work to the single-robot localization problem, is explained in Chapter 3, together with the implementation, test environment details, and the test results in both simulated and real environments. The implementation details and experimental results of the proposed multi-robot localization algorithm, CR-MCL, are discussed in Chapter 4. Chapter 5 presents the comparisons, comments, and discussions on the proposed works. Finally, Chapter 6 presents the conclusion.

2. BACKGROUND

Localization is a challenging field of robotics where many approaches have been introduced. These approaches are generally based on some well known families of solutions. A useful taxonomy and brief introduction to some of the most common ones of these approaches are covered in this chapter.

2.1. Localization Problem

The localization problem is the detection of the position of a robot relative to its environment, using the information about the environment gathered by the robot with its sensors (e.g. infrared, camera, etc.). Unfortunately these sensors and the environment are uncertain (except for specially designed toy problems), so the results are mainly erroneous and inaccurate. Accordingly, localization still remains a nontrivial and challenging problem. The localization problem can be divided into three sub-problems (Thrun *et al*, 2005) as follows:

- **Position (pose) tracking** : This requires keeping track of the robots' position (x, y coordinates and heading) using odometry, with the assumption that the initial position of the robot is known (Fox *et al*, 1999b). Unfortunately as time passes, dead reckoning errors grow cumulatively, and in real time applications the initial position can not be known in most cases. It can be also named as *local* localization since it is based on local uncertainty.
- **Global localization :** This problem requires the robot to find its location in the environment without using *a priori* information. This method generally makes use of the sensors to get information about the environment. Global localization based on only sensory information is a hard task due to the uncertainty associated with the sensors, robot's motion, and the dynamic nature of the environment, and has become a challenging problem for researchers. Therefore, from the simplest geometric calculations which do not consider uncertainty at all, to statistical solutions which cope with uncertainty by applying sophisticated

models, many solutions have been proposed for this problem. Although some of these approaches produce remarkable results, due to the nature of the typical environments they are not satisfactory because fast solutions requiring less memory and computational resources are demanded. This is especially true for a real-time application in a dynamical soccer field using robots with onboard computational resources. Generally, solutions producing precise results suffer from slowness, and high memory usage, whereas a fast solution typically produces only coarse results. Even when they produce precise local results, some approaches like Kalman filters fail to find the global position.

- **Kidnapping :** This is possibly the hardest problem among others. In the kidnapping problem, the robot is taken from its current location and carried to another location by teleportation without the information of the robot. So suddenly, the whole belief set of the robot becomes invalid. Therefore the robot should realize this, and initialize its beliefs from scratch, and relocalize itself in its new position. Since most of the state-of-art localization methods fail in case of kidnapping problem, it is a good indicator in the evaluation of the localization algorithms.

The localization problem can also be classified as *active* and *passive* localization. In active localization, the robot actively searches its environment to find landmarks to localize itself better, and decrease its localization error. So in active localization, the results are more successful. whereas in passive localization, the robot is busy with its task, and it runs the localization as a background task, using the observations it gets during its main task implementation. Notice that, even if the robot gets lost, it would not seek for landmarks to localize itself, unless it is a part of its current task. So the error rate is larger in passive localization.

Another dimension in localization problem is involved with the number of robots taking place in the localization. Only a single robot might localize itself using its actions and observations, or a team of robots might localize themselves using the intra-team shared information, besides the single robot facilities.

The Localization problem is also studied in both static and dynamic environments. In dynamic environments, there might be other moving obstacles, besides the environment could change, so they are noisier, and harder to cope with.

In this work, the passive global localization and kidnapping problems for dynamical environments, which are the hardest tasks among localization problems, are studied. Solutions are proposed for both single and multi robots.

2.2. Single Robot Localization

In this work, first several well-known single robot localization techniques are studied; their advantages and disadvantages are analyzed. In the following sub sections these techniques, which vary from the simplest geometrical approach to hybrid solutions which try to overcome failures by combining or extending the advantageous parts of some well-known techniques, are described briefly (Figure 2.1).

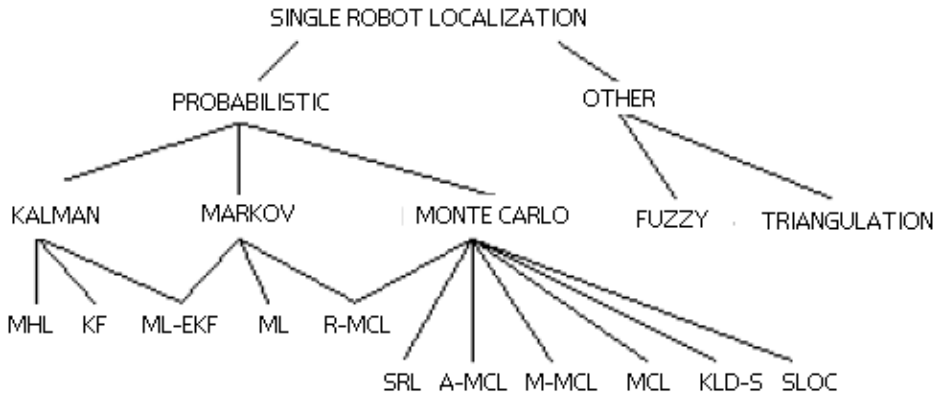

Figure 2.1. The classification of single robot localization methods

These approaches estimate the position of the robot using the observations and odometry. The probability of the robot being in a location l ($p(l)$) is estimated using the following Bayesian update rules:

$$p(l) = \int p(s_n|l)p(l)ds_n \tag{2.1}$$

$$p(l) = \int p(l|a, l')p(l')dl' \tag{2.2}$$

where the robot is given sensory inputs s_n , and executed action a, and l' is the location where the robot is in before the action is executed. Equation 2.1 represent the (*observation update*) and Equation 2.2 represents the (*odometric update*). The methods differ in representing the probability $p(l)$ and other uncertain features like sensor or motion model as briefly analyzed in the following subsections.

2.2.1. Triangulation

The triangulation technique is the simplest localization method and it differs from the other probabilistic methods studied throughout the rest of the chapter since it disregards uncertainty instead of modeling it. This technique uses the geometric properties of triangles based on the distances and/or angle measurements between the robot and the observed landmarks (Betke and Gurvits 1994, Betke and Gurvits 1997, Hightower, 2001).

In the case where both the distance and angle measurements are used, the position of the robot is calculated using the measured distance between the robot and the multiple reference positions. For the position estimation in two dimensions, distance measurements from three non-collinear points are required (Figure 2.2). The angle measurements with respect to the reference points are required to calculate the orientation of the robot.

In the Figure 2.2, and Figure 2.3, d_1, d_2, and d_3 represent the observed distances between the robot and each observed landmark, which are shown by big filled circles. In the Figure 2.3, θ_1 and θ_2 are the angles between the robot and the landmarks.

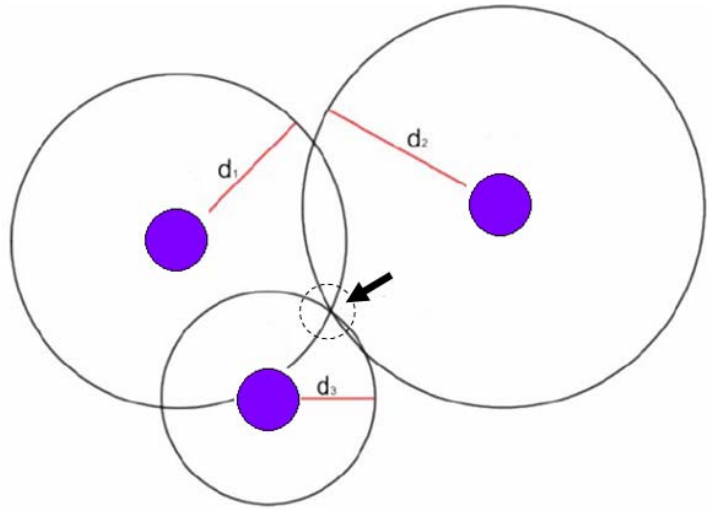

Figure 2.2. The triangulation method with three points

Two angle measurements and one distance measurement can also be used for calculating two dimensional localization by triangulation (Figure 2.3). This case is similar to the former case, except the fact that instead of distance measurements, angle measurements are used for calculating the position of the robot. Unfortunately, in case of noisy data, the error increases drastically. Therefore, whenever more perception data are available, they should be used to reduce the error. As a result of these cases, more than one position estimate could be found. In this case, different combination methods such as averaging can be used to obtain a final position estimate.

In (Betke and Gurvits 1994, Betke and Gurvits 1997), both the cases with perfect and noisy data are aimed to be solved by extensions to triangulation method.

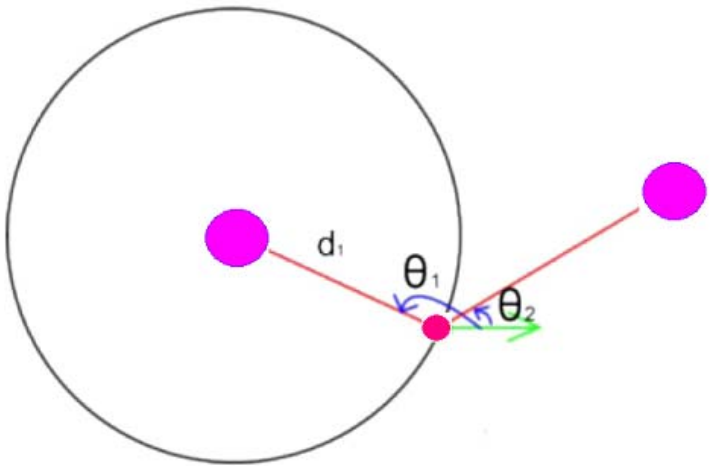

Figure 2.3. The triangulation method with two angle and one distance information

2.2.2. Markov Localization Method

Markov localization (ML) is the most general method since it does not make any distribution assumptions, and allows any kind of distribution to be used. Although this feature makes it flexible, it adds a computational overhead. There are many works based on ML (Fox *et al*, 1999b, Burgard *et al*, 1996, Fox *et al*, 1999c, Thrun *et al*, 2001, Schulz and Burgard, 2001) in the literature. In this work we will consider grid based ML, where the probability of being in a particular grid is represented by a piecewise linear function.

As stated in (Fox *et al*, 2000), ML uses odometry measurements a and perceptional measurements o to estimate the current position of the robot. The robot maintains a belief over its position which is denoted as $Bel^{(t)}(L)$ at time t. The variable L in this representation is a three-dimensional random variable composed of the robot's position and its heading direction. $Bel^{(t)}(L = l)$ is the belief representation showing that the robot is at location l. The initial knowledge of the robot is stated by $Bel^{(0)}(L)$ which is initialized by a uniform distribution, in case of no initial location information. $Bel^{(t)}(L)$ shows the posterior belief with respect to all data collected up to time t as in

$$Bel^{(t)}(L) = P(L^{(t)}|d^{(t)}), \tag{2.3}$$

where $d^{(t)}$ denotes the data collected up to time t. For the perception update, the last item in $d^{(t)}$ is denotes perception data, $o^{(t)}$. Using the Markov assumption, $Bel^{(t)}(L = l)$ is calculated for each l as in Equation 2.4, and it is updated as in Equation 2.5.

$$\begin{aligned} Bel^{(t)}(L = l) &= P(L^{(t)} = l|d^{(t)}) \\ &= \alpha\, P(o^{(t)}|L^{(t)} = l)\, Bel^{(t-1)}(L = l) \qquad (2.4) \\ Bel(l) &\leftarrow \alpha\, P(o|l)\, Bel(l) \qquad (2.5) \end{aligned}$$

here α is a normalizer independent from l. Based on the perception model, and per-

ceived data, $P(o^{(t)}|L^{(t)} = l)$ is the probability of making the perception $o^{(t)}$ given that the robot is at l at the time t; and $Bel^{(t-1)}(L = l)$ is the belief that the robot was at l at the time $t - 1$. For the computation of $P(o|l)$, different sensor models (Fox *et al*, 1999b) could be used.

For the odometry update, the last item in $d^{(t)}$ is an odometry datum, $a^{(t)}$. Based on the motion model, and odometry data, and using the Theorem of Total Probability, $Bel^{(t)}(L = l)$ is calculated for each l as in Equation 2.6, and it is updated as in Equation 2.7.

$$\begin{aligned} Bel^{(t)}(L = l) &= P(L^{(t)} = l|d^{(t)}) \\ &= \int P(L^{(t)} = l|a^{(t)}, L^{(t-1)} = l')\, Bel^{(t-1)}(L = l')\, dl' \qquad (2.6) \\ Bel(l) &\leftarrow \sum_{for\ each\ l'} P(l|a, l')\, Bel(l')\, \Delta l' \qquad (2.7) \end{aligned}$$

The ML method uses a histogram filter for posterior belief. If it uses fine grained grid cells, the algorithm slows down. Otherwise if the grids cells are coarse, this causes information loss, the filter may not even work properly. The most basic version of ML uses time-invariant same sized grid cells. Commonly grid cell size is chosen as 15 cm. The environment is assumed to be static in ML which is not realistic in our case. There are some works that consider this problem (Thrun *et al*, 2005). There is also a problem related with the motion model. Using the motion model on the center of the grid cell yields a poor solution since when the cell size is 15 cm, and odometry update comes in every second for a robot that moves 1 cm/sec, the robot is still in the same cell, after moving several steps, so can not perform a state transition. There are several solutions for this problem. One can modify both the motion-model and measurement by inflating the amount of noise (Thrun *et al*, 2005), but this will reduce the amount of information extracted from the sensors. One can also use the ratio between the moved distance and cell diameter as the probability of moving to a nearby cell. But this would make the robot move to the next cell even with a small motion, so the move is much faster than commanded.

2.2.3. Monte Carlo Localization

Monte Carlo Localization (MCL) is a version of ML that relies on sample-based representation and the sampling/importance re-sampling algorithm for belief propagation (Thrun *et al*, 2001, Schulz and Burgard, 2001). Beliefs are represented by a set of weighed samples (particles) which are of type $((x,\ y,\ \theta),\ w)$, where $w \geq 0$ is a non-negative numerical weighting factor such that the sum of all w is one. The weighting factors are called *importance weights*. Odometric and sensory updates are similar to ML. They are performed within the prediction and correction steps. The algorithms start with a set of uniform random samples S'. At each perception or action, $p(l)$ is calculated according to the following steps (Thrun *et al*, 2001, Lenser and Veloso, 2000, Gutmann and Fox, 2002):

procedure $MCL(S', a, o)$
1: **for** $i = 0$ to n **do**
2: draw by replacement random sample l' from S' according to $w_1..w_n$ $(resampling)$
3: draw sample l' with $p(l|a, l')(sampling)$
4: calculate w'=$p(l'|o)$ $(importance\ sampling)$
5: add (l', w') to S'
6: **end for**
7: normalize the importance factors w' in S'
8: return S'

Figure 2.4. The MCL Algorithm

The MCL algorithm in Figure 2.4 takes the sample set S', observation (sensor readings) o, and action (movement readings) a as input. Whenever an action is done, a new sample l' is drawn according to its probability density $p(l|a, l')$ and replaced with sample l. The probability density functions of motion and observations depend on the robots used, and their capabilities. The models of laser range finders or sonar sensors would be used in perception (Thrun *et al*, 2001), as well as onboard cameras.

In case of a sensory update, the samples are given weights equal to their probability given the sensor reading. Using these weights, new unweighted samples are then drawn from the sample set by replacement. The probability of drawing a sample is

proportional to the weights. Next, these samples are added to the sample set. There is no addition of new samples in this step.

Notice that MCL represents the probability distribution of location l by a sample set. The samples are located such that, the population size of samples is proportional to the probability of the robot being in that location and the surroundings. After a couple of steps, the samples converge to some place in the environment. The position of the robot is found by taking the average of the positions of the samples, and the standard deviation gives the uncertainty of the robot. So this fact could also be used to dynamically change the number of samples. If the robot is totally unaware of its place, more samples could be used to represent the field. On the other hand, if the uncertainty is low, then a smaller number of samples is sufficient for finding the robot's location. The original MCL algorithm does not work well in case of kidnapping. So several MCL extensions were proposed to overcome this problem. These extensions add new samples to the sample set from different parts of the field, and differ in the number of new samples and when to add these samples. Some of these methods are Sensor Resetting Localization (SRL), Mixture MCL (Mix-MCL), Adaptive MCL (A-MCL), and Kullback-Leibler Distance (KLD)-Sampling .

In SRL, when the likelihood of the current observation is below a threshold, a small fraction of uniformly distributed random samples is added (Lenser and Veloso, 2000).

Mix-MCL additionally weights these samples with current probability density $p(l)$. This method has been developed for extremely accurate sensor information (Gutmann and Fox, 2002).

Adaptive MCL only adds samples when the difference between short-term estimate (slow changing noise level in the environment and the sensors) and the long-term estimate (rapid changes in the likelihood due to a position failure) is above a threshold (Gutmann and Fox, 2002). This approach was applied to the Robocup domain in 2002 by the University of Washington team (Crisman *et al*, 2002). The key idea is to use a

combination of two smoothed estimates (long term and short term) of the observation likelihoods (Gutmann and Fox, 2002).

In KLD-sampling method, the size of sample sets are time-variant, which increases the efficiency, especially if the complexity of the beliefs vary drastically over time (Fox, 2003).

As stated before, when the robot is unaware of its location, it needs more samples to be located. But as the number of samples increase, the algorithm dramatically slows down. Besides if not enough landmarks are observed, after some steps, the robot could converge to some location with a small number of distinct samples, and would not move to the correct location, after new observations. MCL is capable of handling only small systematic errors. When errors get bigger the overall localization error gets cumulatively larger and the algorithm is not capable of overcoming this situation. The algorithm also can not handle errors due to unexpected movements. The time to recover from an error is proportional to how big the error is (Lenser and Veloso, 2000). Even with adaptive sample size not all of the problems can be solved.

2.2.4. Kalman Filter Method

Kalman filter (Kalman, 1960, Maybeck, 1990, Welch and Bishop, 2006) is a well-known approach for the localization problem. It implements belief computation for continuous states. It is similar to Markov Localization but it makes a Gaussian distribution assumption. This filter integrates uncertainty into computations by making the assumption of Gaussian distributions to represent all densities including positions, odometric and sensory measurements. Position estimates are updated by odometry and sensing alternately using the property that Gaussian distributions can be combined using multiplication (Stroupe and Balch, 2002, Stroupe *et al*, 2003).

It is unimodal, it can posses only a single maximum. Therefore only one pose hypothesis can be represented. So the method is unable to make global localization, and can not recover from total localization failures (Gutmann and Fox, 2002). Therefore

usually, this method is used to track the object's pose locally, together with another method, which is responsible for global localization making up a hybrid approach. It is a very efficient algorithm, giving precise results, whereas, additional computation of coefficient and covariance can increase the computational time undesirably (Stroupe *et al*, 2003).

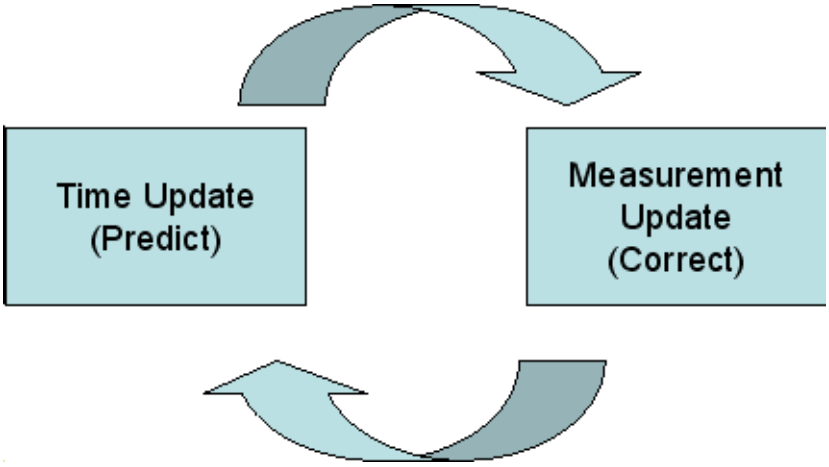

Figure 2.5. KF method

procedure $TimeUpdate$(Initial estimates for $\hat{x}_{k-1}$ and P_{k-1})

1: Project the state ahead
$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$
2: Project the error covariance ahead
$P_k^- = AP_{k-1}A^T + Q$
3: return $\hat{x}_k^- and P_k^-$

Figure 2.6. The KF Time Update Algorithm

procedure $MeasurementUpdate$

1: Compute the Kalman gain
$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$
2: Update the estimate with measurement
$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$
3: Update the error covariance
$P_k = (I - K_k H)P_k^-$

Figure 2.7. The KF Measurement Update Algorithm

Beliefs are represented by multivariate normal distributions and characterized by first and second moments, namely mean and covariance. Different notations are used in the literature, in this section the notations in (Leonard, and Durrant-Whyte, 1991) are used to keep parallel with the figures. We have two basic states in the cycle, *Time*

update (predict) and *Measurement update(correct)* as in Figure 2.5. In the Time update (Figure 2.6), the current state estimation is predicted, and in the Measurement update (Figure 2.7), it is corrected using the current measurements. The state transition function should be linear with Gaussian noise. In the first step, the matrices A and B satisfies this linearity, and u_k is the control vector at time k. The algorithm takes mean and covariance at time $k-1$, and estimates their values at time k. The predicted belief is transformed to desired belief in correct state, with the help of *Kalman Gain*. Kalman Gain specifies the degree to which actual measurement z_k is integrated into the new estimate. The difference between the actual measurement and the expected measurement Hx_k is the *Innovation*. Finally, the new covariance is calculated using the information gain.

If the system model is non-linear and potentially numerically unstable, Extended Kalman filter (EKF) is used. Here the linearity assumption in state transitions and measurements is relaxed and these are governed with non-linear functions as shown in Figure 2.8 and Figure 2.9.

procedure *TimeUpdate*(Initial estimates for $\hat{x}_{k-1}$ and P_{k-1})

1: Project the state ahead
$\hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1}, 0)$
2: Project the error covariance ahead
$P_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T$
3: return $\hat{x}_k^- and P_k^-$

Figure 2.8. The EKF Time Update Algorithm

procedure *MeasurementUpdate*

1: Compute the Kalman gain
$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + V_k R_k V_k^T)^{-1}$
2: Update the estimate with measurement
$\hat{x}_k = \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-, 0))$
3: Update the error covariance
$P_k = (I - K_k H) P_k^-$

Figure 2.9. The EKF Measurement Update Algorithm

2.2.5. Multi Hypothesis Localization

The Multi Hypothesis Localization (MHL) method discussed in (Kristensen and Jensfelt, 2003) aims to avoid problems caused by using a single Gaussian, by considering a mixture of Gaussians, thus enabling the representation of any given probability distribution of the robot pose.

2.2.6. Markov Localization - Extended Kalman Filter

Markov Localization - Extended Kalman Filter (ML-EKF) method is a hybrid method aiming to make use of the advantages of both methods, taking into consideration the fact that ML is more robust and EKF is more accurate. So this method finds the location of the agent coarsely by grid based ML and then inside this area uses EKF to find a more accurate solution (Gutmann and Fox, 2002, Gutmann, 2002).

2.2.7. Fuzzy-Localization

In (Buschka *et al*, 2000, Saffiotti, 2000, Saffiotti *et al*, 2002), a different approach based on fuzzy logic is introduced and implemented in the Sony legged robot league (AIBOs). This is a grid-based approach that presents uncertainty in terms of fuzzy membership functions. The range r and the heading θ of the robot are represented in terms of fuzzy sets. The trapezoidal membership functions are useful in representing the uncertainty in sensor readings, and bias is useful in recovering from kidnapping problem. All trapezoidal fuzzy sets, are represented with tuple (θ, Δ, α, h, b), where θ is the center, Δ is the width of the core, α is the slope, h is the height and b is the bias, as in Figure 2.10 (Buschka *et al*, 2000).

Bias is used to integrate the idea that "the solution could be somewhere else". The method does not critically rely on the accuracy of these parameters (Buschka *et al*, 2000). This is a grid-based approach, and the distance from the observed landmark to every grid is represented by tuples of the form (r, Δ, α, h). The robot's position at time t is represented by a two-dimensional fuzzy grid map Gt, where $Gt(x, y)$ measures

the degree of possibility. For sensory updates, the possibility distribution $St(x, y|r)$ measures the probability of the robot being in (x, y) with the information that the observed landmark is at distance r from the robot.

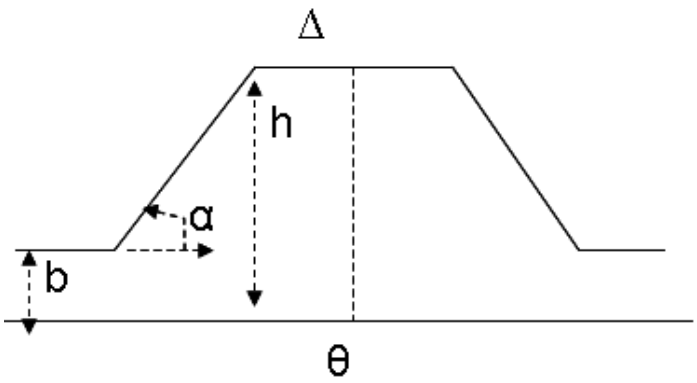

Figure 2.10. Trapezoidal fuzzy sets

The approach follows a predict-observe-update cycle. In the observe step, the sensory information (observed range) is converted to possibility distribution of the grids, as described in the previous paragraphs. The predict cycle was not properly implemented until 2002. This step makes use of the odometric information to update the current position. Previously, this step was implemented like blurring from the grid to all directions by a maximum amount that could be achieved by the robot (Buschka *et al*, 2000). After 2002, the locomotion module of the UNSW team was used by TEAM SWEDEN, so it could be possible to use odometric information (Saffiotti *et al*, 2002). Lastly in the update part this information is integrated into the fuzzy grid map, by a fuzzy intersection operator (Buschka *et al*, 2000).

2.2.8. Simple Localization

The Simple Localization (S-LOC) method is a technique which represents every perception by a sample. The old position estimation is also represented by a sample. This sample set together with a history based module, is used to estimate the current position of the robot (Çelik, 2005, Köse *et al*, 2006).

In S-LOC, assuming that the robot has the previous pose position, the orientation

of the sample pose is calculated so that the perception would be on the correct direction. The position of the sample pose is then calculated on a line through the old position and along with the same direction assuming that the perception was exact. These temporary pose samples are calculated for each perceived landmark. In addition to them, the previous pose is also used as an additional sample.

For each sample pose, the likelihood is calculated. Assuming that the robot's actual pose is the sample pose being processed, the Euclidean difference of the perceived landmarks' positions and their actual positions is calculated. Together with the confidence of the perception from which that sample pose is calculated, these differences are used in the calculation of the likelihood of that sample pose. For the sample that is copied from the previous pose, instead of the confidence of the perception, the confidence of the previous pose is used.

The likelihoods of the sample poses are used for calculating their weights, and a new pose position is calculated as the weighted average of these sample poses' positions. This weighted average pose position is then used together with the previous pose estimate's position to calculate the current pose estimate's position. The purpose of not using the weighted average pose directly is to keep a history in order to prevent fluctuations of the pose estimate and make it more stable. After the current position of the agent is estimated, the current orientation of the agent is calculated using the current position estimation and the perceptions.

In the case of having no perception at a certain time, the current pose estimate could be obtained by decreasing the confidence of the previous pose estimate.

The odometry update process is as simple as updating the pose estimation with the odometry data. Since only the pose estimation is used from the previous perception update, no additional update or calculation is necessary.

In a way, S-Loc is similar to MCL as the sample poses are used in the same way they are used in MCL. The main difference is the selection of these sample poses.

In MCL, there is a large number of pose samples, and they are populated according to their confidences, and randomly mutated for small changes. In S-Loc, new pose samples are calculated at every estimation, and for each perceived landmark a pose sample is calculated. This way S-Loc becomes a much lower cost, i.e. much faster, localization method with an accurate pose estimation capability. The memory used in S-Loc increases the robustness of the system even further and the big jumps of the pose estimate are prevented.

2.3. Multi Robot Localization

The use of cooperative robotics is becoming more prominent in many key application areas. Multi-robot teams where robots cooperate with each other, and/or with human beings become popular as their performance are shown to be better, more reliable and more flexible than single robots, in a variety of tasks (see (Saffiotti, 2002, Ribeiro and Saffiotti, 2002)) for many papers about cooperative robotics). Unfortunately there is yet a limited number of applications in this area, and many of these are about toy problems or limited implementations of applicable problems due to the difficulty of the problem. Problems in the coordination of the robots, efficient usage of limited resources and the communication burden discourages researchers to work on real-time problems with dynamic environments. Robot soccer is such an environment, with its real-time, complex and dynamic nature, and, implementation of cooperative robotics to legged robots makes it even more challenging where locomotion (moving the legged robots) becomes a real bottle-neck.

One of the fundamental goals of this work is to solve multi-robot localization in an efficient manner. Global collaborative localization in autonomous mobile robots is a highly challenging task, which is generally more successful than single robot localization, despite its high complexity and associated communication problems. It is a challenging field of robotics where many approaches have been introduced. These approaches are generally based on some well known families of solutions. This problem is usually solved by localizing each robot in the domain individually. Some works use the leap frog method where some robots stand still to serve as landmarks for others,

and some works only solve the relative localization problem.A useful taxonomy and brief introduction to some of the most common ones of these approaches are covered in this section.

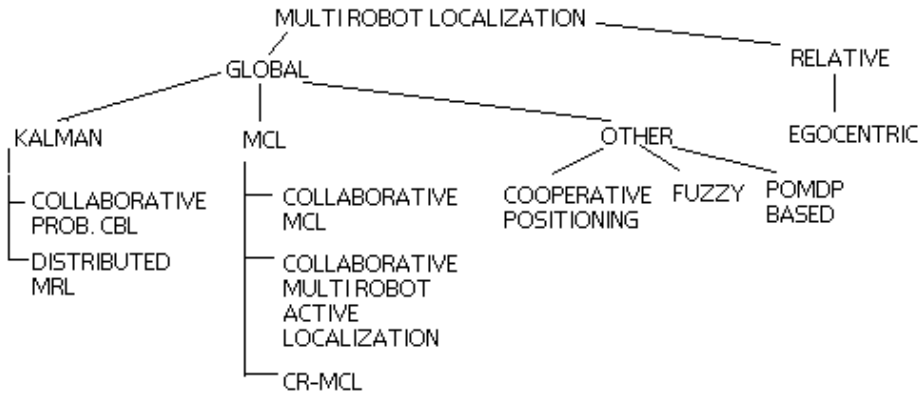

Figure 2.11. The classification of multi robot localization methods

2.3.1. Collaborative Probabilistic Constraint-based Landmark Localization

There are several Kalman based approaches in this domain. The work in (Stroupe and Balch, 2002) uses a Kalman filter based approach to find the global locations of the robots in the team. In this work, distance and bearing information are used separately, each in a separate cycle. The observation of the other robot related to the robot itself is shared and used whenever two robots meet. In the passive localization part, the robots use each other as landmarks. This work uses action and sensory information as separate Kalman filters.

2.3.2. Distributed Multi-Robot Localization

In (Roumeliotis and Bekey, 2000, Roumeliotis and Bekey, 2000, Roumeliotis and Rekleitis, 2004) a Kalman filter based approach is used to combine multi-robot information. It is assumed that the robots can detect and identify each other. They have sensors to detect objects around and their own motion to use in localizing themselves. The first approach is to use centralized Kalman filter to localize the team. The second approach uses M different reduced dimension Kalman filters one for each robot to

localize the team. Only if the robot detects others it shares information. When two robots detect each other they share their estimations. In these works it is claimed that this is the first multi robot localization approach to explicitly address the problem of sensor data interdependencies that appear when robots exchange information regarding their pose estimates. The cases where when one robot is stationary, when nobody has absolute position estimate, and when one has absolute position estimate, are tested with real robots, and the results are compared with the dead-reckoning error.

2.3.3. Cooperative Monte Carlo Localization

In (Fox *et al*, 2000, Fox *et al*, 1999a), probabilistic methods are used to synchronize each robot's belief whenever detection takes place. In this work the following assumptions are made:

- Robots can detect and identify each other
- No negative example-robots always can detect each other
- The map is known (indoor environment).
- Each robot can localize itself with action and sensor information which is available for it, if it can not detect anybody
- The robots can not share information unless a predefined time passes between two sequential detections.

This work approximates sample sets using piecewise constant density functions represented by a tree. When a robot detects another, it forms a density tree from the sample set it produced using the detected data, and then shares this tree with the detected robot. Each node in the tree is represented with a hyper-rectangular subspace of the 3-D state space of the robot. Initially all samples are in the root node. Recursively each node is split until a termination condition is fulfilled. The more samples exist in a region, the finer-grained the tree representation. After the tree is grown, the density of each leaf is calculated using the quotient of the sum of all the weights of all samples in the leaf, divided by the volume of the region covered by the leaf. This is the maximum likelihood estimation of (piecewise) constant density

functions (Fox *et al*, 2000). Then the samples are transformed to a density tree. Then these density values are multiplied into each individual sample of the detected robot n. At the end a refined density for robot n is produced. The same can be applied to robot m.

procedure *Multi robot Localization*

1: **for each** location l **do** /*initialize the belief*/
$Bel_n(l) \leftarrow P(L_n^{(0)} = l)$
2: **end for**
3: **forever do**
4: if the robot receives new sensory input o_n do
5: for each location l do /*apply the perception model*/
$Bel_n(l) \leftarrow \alpha\, P(o_n|l)\, Bel_n(l)$
6: **end for**
7: **end if**
8: **if** the robot receives new odometry reading a_n do
9: **for each** location l **do** /*apply the motion model*/
$Bel_n(l) \leftarrow \int P(l|a_n, l')\, Bel(l')\, dl'$
10: **end for**
11: **end if**
12: **if** the robot is detected by the m-th robot do
13: **for each** location l **do** /*apply the detection model*/
$Bel_n(l) \leftarrow \int P(L_n = l|L_m = l', r_m)\, Bel_m(l')\, dl'$
14: **end for**
15: **end if**
16: **end forever**

Figure 2.12. Localization algorithm for multiple robots

In Figure 2.12 (Fox *et al*, 2000), the update rules for action and observation for multi robots are summarized. There, $Bel_n(l)$ represents the belief of a robot for being in location l. $P(o_n|l)$ is the perception model such that it gives the probability of making observation o by robot n, at location l. $P(l`|a_n, l)$ is the motion model such that it gives the probability of reaching to location $l`$ by making action an by robot n, at from location l. When a robot m observes another robot n, its own belief about n and n's current belief are used to estimate the new belief of n. The algorithm is based on the assumptions that there are no negative sightings (robots can always see each other) and robots can detect each other.

2.3.4. Cooperative Positioning System

In (Kurazume and Hirose, 2000), the authors assumed that there is at least one stable robot, at a time, which serves as a landmark to the moving robots. This is called Cooperative Positioning System (CPS). This can be used in outdoor environments, the map need not be known, and the robots do not need other landmarks. They assume the robots can detect each other and measure the relative distance between themselves and the detected robots. The robots are divided into two groups, one of the groups remains stationary and observes the others, while the others move. Then the stationary ones share the information related to the move of the other group with them, so that they can localize themselves. Then the stationary group moves, and the other group stays stable and serves as landmarks. This continues until the robots reach the target place.

2.3.5. Robust Multi-robot Object Localization Using Fuzzy Logic

The work in (Canovas *et al*, 2004) maintains a consensus between robots instead of trade off provided by taking average. Information is combined by Fuzzy information fusion. It is claimed that fuzzy fusion is better than similar approaches since it provides a consensus, not a trade off, and it automatically discounts unreliable information. This is a grid based approach, and the degree of possibility that object is located at grid x given the available information coming from the sensors and action information is held. When needed, the point estimate is found by center of gravity. The robots do not share point estimates but the whole distribution, when they detect each other. The overload of transmitting this large amount of information is being compensated by converting cell values to one byte and treating the grid as gray-scale image, then using run length encoding. The ball grid is sent only if the information is new or with better quality than last sent.

2.3.6. Collaborative Multi-Robot Active Localization

In (Jones and Shel, 2004) "active localization" issue is considered. In collaborative multi-agent localization, the robots use MCL to localize themselves. When a robot detects another robot, the detecting robot shares a subset of its sample set with the detected robot. In case of active localization, when a robot fails to localize itself, it calls for help. Another robot which localizes itself accurately, comes to help and helps the failed one to localize itself. If it can not come on time or help, the failed one tries to localize itself or calls another one to help. They use high level actions e.g. spin around itself, stop, etc. The following assumptions are made:

- Robots can detect and identify each other
- No false detection or incorrect identification
- Map is known-indoor environment
- All robots see another robot to update their believes
- The range and bearing are modeled with Gaussian noise

2.3.7. Representing Hierarchical POMDPs as DBNs for Multi-scale Robot Localization

The Hierarchical partially observable Markov decision processes (H-POMDPs) are represented as Dynamic Bayesian networks (DBNs) in (Theocharous *et al*, 2004). In particular, they focus on the special case of using H-POMDPs to represent multi-resolution spatial maps for indoor robot navigation. They use basic actions like move forward, and move backward, and represent pose and heading as nodes.

2.3.8. Ego-Centric Approach

This is a cooperative method for relative localization of mobile robot teams, where the global positions of the robots are discarded, but every robot in the team holds separate sample sets for the rest of the team, and estimates the relative positions of every other robot using a Mixture-MCL based approach (Howard *et al*, 2003).

3. PROPOSED METHODS FOR SINGLE ROBOT LOCALIZATION

In this work, we developed several single robot localization methods. The most successful one is the R-MCL which is one of the outstanding methods in this domain. It is a hybrid method based on the ML and MCL methods. In this chapter, a detailed description and analysis of the methods are presented. The R-MCL method is tested on both a simulated environment and on real robots, and comparison of results with the other outstanding methods in the same domain is also given.

3.1. Geometrical Localization

The geometrical localization method assumes the input data is measured exactly (does not contain noise), and therefore does not need any error modeling. Our previous algorithm in (Akın *et al*, 2001) required at least two landmarks to be seen at any time to calculate the position accurately. Although it worked also for the one landmark case, it could not give satisfactory results. This new method is designed to work with one landmark information which is much more realistic within the new field sizes. So even if the robot sees more than one landmark, they are treated separately and one-landmark information is used at each step. The ratio of the distance between the predicted location and the observed location of the landmark is used to predict the new x and y coordinates of the robot as in Equation 3.1 and Equation 3.2 (Stroupe *et al*, 2003). In these equations, x' and y' represent the new x and y coordinates, X_L, and Y_L are the coordinates of the observed landmark, and X_O and Y_O are the coordinates of the old position of the robot. As shown in the (Figure 3.1), d is the calculated distance between the landmark and the old position of the robot, and r is the currently observed distance between the landmark and the robot.

$$(X_L - x')/r = (X_L - X_O)/d \tag{3.1}$$

$$(Y_L - y')/r = (Y_L - Y_O)/d \tag{3.2}$$

The bearing is also found by using the new predicted x and y coordinates. Whenever new visual data come, the new position is calculated based on the measurement and the old position. A point between the newly measured position and the old position is taken as the new position. This new position is placed between the two positions proportional to the belief of the robot in them. The assumption here is as follows: The more you believe in a position the closer you are to that position. This is used to reduce the effect of inaccurate measurements on the new position. When the odometric data arrive, the position is blurred among the moved distance and heading. The bearing is added to the original heading and it is normalized to give the new heading of the robot. The odometric data consist of the distance moved forward, left and the bearing of turn. This method assumes that the measurements are exact, or noise is below a threshold.

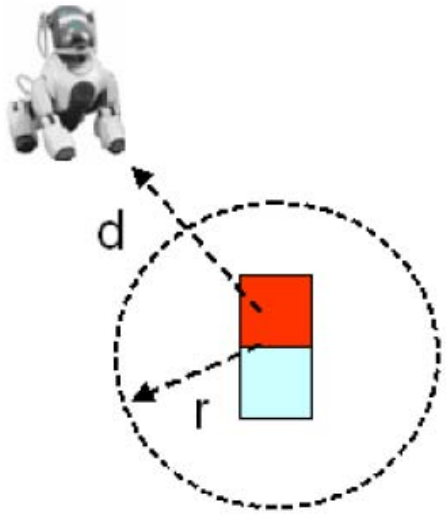

Figure 3.1. Calculation of distance to an observed landmark.

3.2. Reverse Monte Carlo Localization Method

As stated in the previous chapter in detail, both the ML and MCL methods are well-known methods and they have a wide range of usage. Some of the advantages of the ML and MCL methods are summarized below:

- they can process raw sensor measurements, there is no need to extract special features,

- they are non-parametric, no unimodal distribution like EKF is needed,
- they can solve global localization, and in some instances the kidnapped robot problem.

ML is a grid based method which allows any distribution to be used to integrate uncertainty in the measurements. It is flexible, robust and converges fast, but is coarse and computationally complex. On the other hand, sample based MCL is not as computationally complex as ML, and gives accurate results. It is typically preferred since its implementation is relatively easy. However, it cannot converge to a position as fast as ML, especially in the case of an external impact on the position of the robot (such as kidnapping). In addition, the number of samples to be used is generally kept very high to cover all the space and converge to the right position. Several extensions have been made for adaptive sample size usage, but these still do not solve the slow coverage problem.

This work uses 2-D Grid based Markov Localization. From now on 2-D Grid based ML is intended when we use the term ML. We do not prefer 3-D Grid based ML since the state space grows dramatically, and slows down the method. So we exclude the orientation information in ML. As the grid cell sizes get smaller, the resultant position becomes more accurate. But since every sensory and action update is applied to each grid cell, this brings a computational overhead to the robot. Besides since it is 2-D, the orientation information which could be vital in case of inadequate observation during a real-time application like soccer game, can not be used at all. In addition, action information can not be used in case of large sized cells, where the actions are very small compared to the cell size, so it really does not make a significant difference when applied as action update. This algorithm enables us to make use of the robustness, and fast coverage from kidnapping facts of ML. Besides by using MCL, we aim to refine the resultant location, and making use of orientation information, and action readings as well.

We propose the Reverse Monte Carlo Localization(R-MCL) algorithm to benefit from the advantages of these two methods while avoiding their disadvantages. The

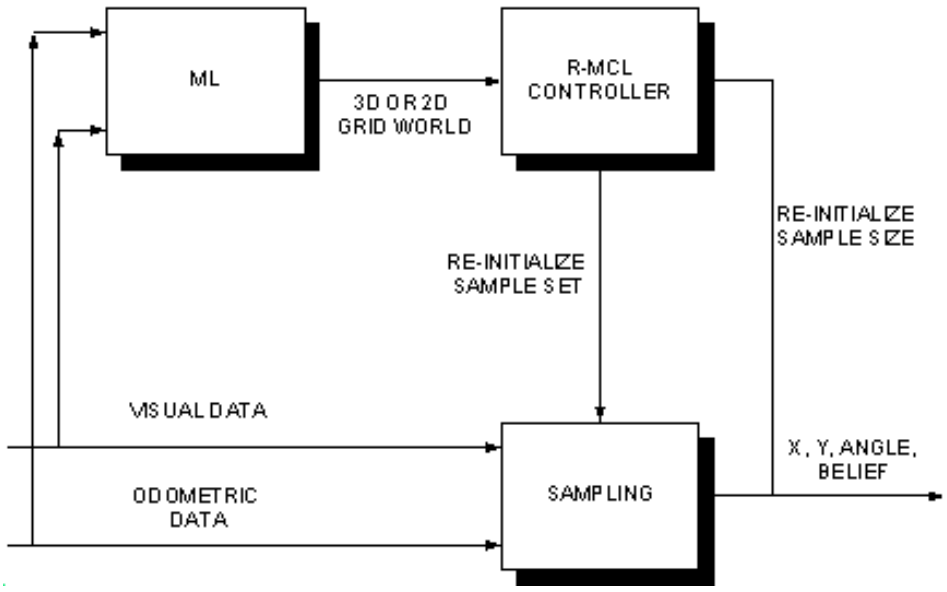

Figure 3.2. The R-MCL working schema

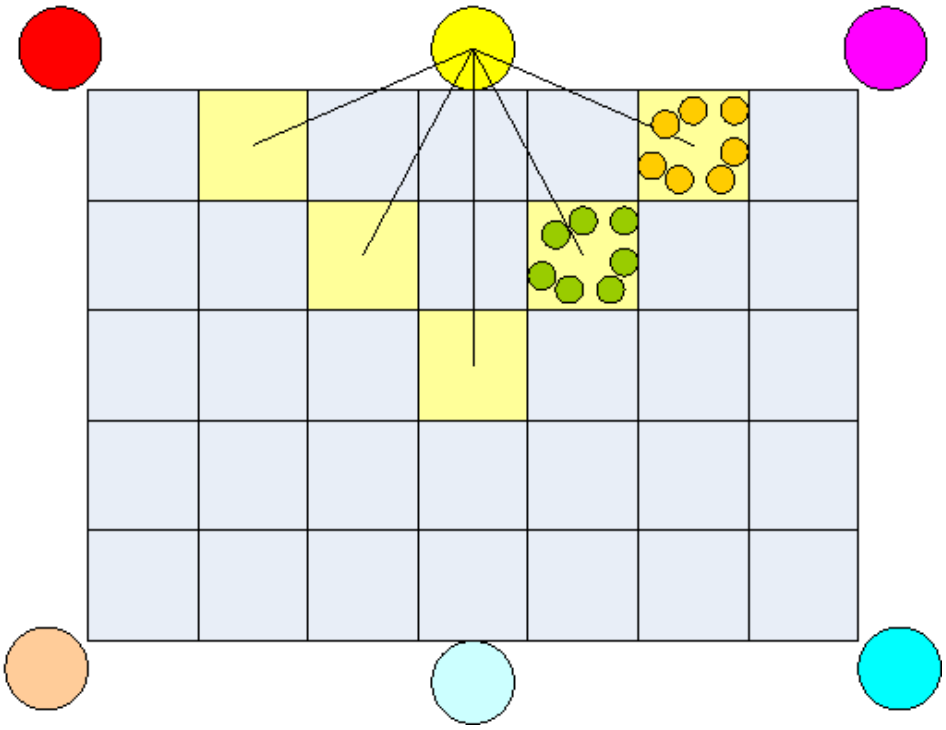

Figure 3.3. The basic representation of R-MCL in the game field

algorithm is called *Reverse* since in the normal MCL methods, first the samples are thrown, then they converge to a place in the field, then the position is estimated. In R-MCL, first the place where the samples should converge is found, then the samples are thrown to find the final position, so the routine is reversed. The average of the thrown samples gives the final position estimate and the standard deviation gives the uncertainty of the final position as in the MCL based methods. In the original MCL, the number of samples is increased to decrease the bias in the result. In R-MCL since we converge by selecting the cells with maximum probability, the bias is already relatively low. (Köse *et al*, 2004, Köse and Akın, 2005, Köse *et al*, 2005, Köse *et al*, 2006, Köse and Akın, 2007).

```
procedure R − MCL(max_grid_array, bool_ML)
 1: if bool_ML==TRUE then
 2:    ML_update
 3:    if ML_number_of_grid_cells_in_max_grid_array < Th_ML then
 4:       MCL_init(ML_samples)
 5:       bool_ML=FALSE
 6:    end if
 7: else
 8:    MCL_update
 9:    MCL_init(ML_samples)
10:    if MCL_lost()==TRUE then
11:       ML_reset()
12:       bool_ML=TRUE
13:    end if
14: end if
```

Figure 3.4. The R-MCL Algorithm

The R-MCL algorithm is given in Figure 3.4. Here, $bool\_ML$ is a boolean variable indicating whether or not to call ML. If it is $TRUE$, ML, otherwise the MCL module is active. The $ML\_reset()$ function initializes all grid cells with equal probability that adds up to 1 for all cells. ML_update works like the normal ML sensory and action update as stated in the previous chapter. After each step the cells with non-zero probability are put into a data structure called max_grid_array. If the number of cells in this structure is smaller than a threshold called th_{ML}, this means the cells have converged to a coarse location in the field. The $MCL\_init()$ function is then called with samples generated by ML as input. Instead of generating random samples, we produce

smaller-sized grid cells from the chosen non-zero weighted grid cells.[1] The weights for each sub cell which is a member of the sample set of MCL now, are calculated next, using the sensory and action readings, and normal MCL updates are done as stated in the previous chapter in detail. Since *bool_ML* becomes $FALSE$, ML is not called anymore, and *MCL_update* function is used. If the uncertainty in MCL is below a threshold Th_{MCL}, this means either the cumulative error increased so much that the robot feels totally lost, or it is kidnapped. The *MCL_lost*() function then returns the value $TRUE$, so the *ML_reset*() function which initializes variables of ML module, such as grid cell weights and *max_grid_array* is called. In addition, *bool_ML* becomes $TRUE$, so in the next step the ML module would be active.

At each iteration the odometric data $(\Delta x, \Delta y, \Delta\theta)$ are used to update the *pose* which represents the current position of the robot, and the weights of the grid cells as given in Figure 3.5.

procedure $ML - Motion(Pose, G, a)$
1: update Pose using motion parameters
2: $Pose_x = Pose_x + \Delta x \times sin(Pose_\theta) + \Delta y \times cos(Pose_\theta)$
3: $Pose_y = Pose_y + \Delta y \times sin(Pose_\theta) - \Delta x \times cos(Pose_\theta)$
4: $Pose_\theta = Pose_\theta + \Delta\theta$
5: update w_i where Pose is in G_i
6: return $Pose, G$

Figure 3.5. The ML Motion Update Algorithm

procedure $ML - Vision(Pose, G, o, m)$
1: $G' = 0$
2: **for** $i = 1$ to n **do**
3: $\quad w_i = vision\_update\_model(G_i, o, m)$
4: $\quad if\ w_i \geq Threshold$
5: $\quad G' = G' + (G_i, w_i)$
6: **end for**
7: normalize w
8: return G'

Figure 3.6. The ML Vision Update Algorithm

The grid cells with weights $w_i \geq Threshold$ are stored in *max_grid_array* (G') and returned from *ML-VisionUpdate* algorithm given in Figure 3.6. The total number

[1]The size of those sub cells can be made adaptive to be used as adaptive sample set size.

of cells is n. The algorithm takes grid array G, map m, observations o, and old pose $Pose$. If the number of grid cells in $G' \geq ML_{Th}$ as mentioned in the R-MCL algorithm in Figure 3.4, then the *MCL Vision Update* algorithm is used to update the sample weights as in the MCL algorithm given in Figure 2.4. The grid cells in G' are divided into smaller cells and the cell centers of these sub cells are added to the sample set as new samples. Currently the number of samples are fixed, but adaptive number of samples could also be used to decrease the computational complexity, and memory requirements, and make the algorithm faster. These samples are then sent to the MCL module, and until the robot is kidnapped and gets lost, the MCL module works with this sample set.

In the implementation of R-MCL in this work many *vision update models* based on the sensor model are used. In Figure 3.7 a simple model used in the tests is presented. In this figure, d_i is the absolute difference between the calculated and the observed distance between the robot's last calculated position and the observed landmark. In the MCL method, not only the distance, but also observed and calculated angle measurements are used to calculate the probability. Different models were analysed and lastly, a similar but more complicated model based on a sigmoid function instead of simple rectangular model (Figure 3.7) is used in the implementation, since it provides higher accuracy in MCL.

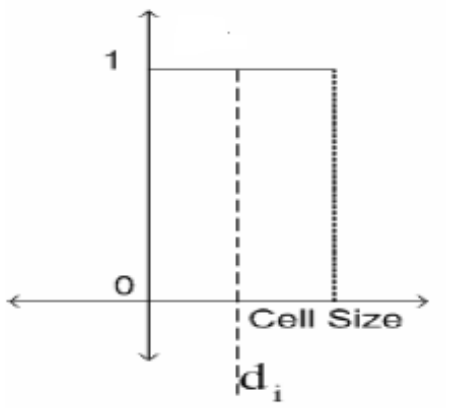

Figure 3.7. The ML vision update model

The MCL method differs from the ML module by the motion model it uses. In Equations (3.3), (3.4) and (3.5) the new (updated) coordinates and orientation of the pose estimate is calculated.

$$PE_x^* = PE_x + \Delta x \times sin(PE_\theta) + \Delta y \times cos(PE_\theta) \tag{3.3}$$

$$PE_y^* = PE_y + \Delta y \times sin(PE_\theta) - \Delta x \times cos(PE_\theta) \tag{3.4}$$

$$PE_\theta^* = PE_\theta + \Delta\theta \tag{3.5}$$

where PE_x^*, PE_y^* and PE_θ^* are the updated x-coordinate, y-coordinate and orientation of the pose estimate; PE_x, PE_y and PE_θ are the x-coordinate, y-coordinate and orientation of the pose estimate before the odometry update; Δx, Δy and $\Delta\theta$ are the odometry data giving the change in the x-coordinate, y-coordinate and orientation.

Notice that, in ML module, 2-D grids are used, so that the orientation information is not taken into consideration, whereas the samples used in the MCL module, also have orientation, so that this valuable information coming from the observation of beacons could be used in position estimation. Especially in cases, where there is high sparsity and noise, orientation measurement which is more reliable than distance measurement plays an important role in position estimation.

This algorithm enables us to make use of the robustness, and fast coverage from kidnapping facts of ML. Besides by using MCL, we aim to refine the resultant location, and making use of orientation information, and action readings as well.

3.3. Fuzzy R-MCL

In this method, the uncertainty model μ_1 which is used in both ML and R-MCL is replaced by the model μ_2 based on a fuzzy membership function in Figure 3.8. The old model was simple and fast but it was not flexible enough to improve success in high sparsity and noise. Especially if the cell size is kept high as in (Köse *et al*, 2004, Köse and Akın, 2005) -30 cm- relative to the works in (Gutmann and Fox, 1998) -5 cm-

a more flexible model is needed to weight the probability of being in that cell. It is not preferable to give same weight to every point in the cells of big sizes, and to the samples inside these cells. Besides fuzzy membership functions also allow us to adapt parameters due to noise and sparsity level, and add bias, when necessary to cope with kidnapping problem.

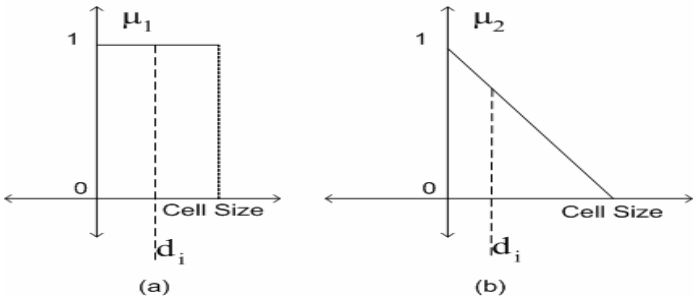

Figure 3.8. The fuzzy membership functions

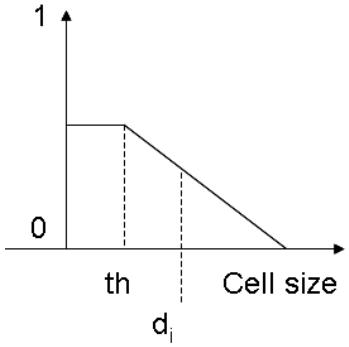

Figure 3.9. The trapezoidal membership function

In both models, d_i represents the difference between the observed relative distance from robot to the currently observed landmark, and the calculated distance from the current cell center to the currently observed landmark. This enables us to weight the samples according to their fitness to the observation and odometry. Several membership functions (e.g. trapezoidal) and different sizes (e.g. twice the cell size) were tested. The triangular model μ_2 is presented in Figure 3.8b. Detailed information about Fuzzy R-MCL with triangular model is found in (Köse *et al*, 2005).

In Figure 3.9, the trapezoidal model is presented. The absolute value of the distance d_i is taken. The threshold th represents the limit value where the cell will get weight 1. The d_i values between th and cell size get values calculated from the

trapezoidal model, and the rest of the cells get 0. There is no bias in this model. The results of this model are presented in the Tests and Results section.

3.4. Tests and Results for Single Robot Localization

In this study, we first compared R-MCL with several methods which are developed to be used in robot soccer, by using a well-known data set in an offline manner. Then we tested the R-MCL method and other methods developed for our soccer team on the real robot, based on a challenging scenario, with varying light conditions. These real-time tests were implemented using current field conditions to test the success and robustness of the method in the real environment, since simulation results may underestimate the problems and bugs related to real life conditions.

3.4.1. The Testing Environment

This work is a part of the Cerberus Team Robot soccer project, and aims to localize the legged robots in the soccer field globally within the rules of Sony Legged Robot League. The Sony Legged Robot League is an international robot competition that has been launched within the RoboCup initiative (Robocup Organization, 2005, Sony Four-Legged Robot League, 2005). In robot soccer, teams of robots, that are capable of seeing and moving, play matches against each other for fixed time periods, and the team with the highest goal score win the match like in real soccer. In order to do this, the player robots must detect their location, the goals, the ball, the members of their team and the opponent team members (optional for high level planning), and place the ball in the opponent team's goal to score. A robot is typically expected to find its own location using the artificial landmarks called *beacons* in the field, and then use this information to find the location of the ball and goal. So localization is a vital problem for robot soccer.

There are several limitations and assumptions related to the rules of the Robocup (Robocup Organization, 2005). The locations of the beacons are given as prior information to the robots. The robots can identify the beacons and estimate their relative

distance to these beacons using their cameras with some noise due to the imperfect vision system and dynamic nature of the world around. With a well developed locomotion module, also odometric data are available to the robot. Consequently, a robot can calculate to which direction relative to its current heading, and how much it has moved during a specific time interval. When this information is not available, a motion model may be used for estimating this, or it may be totally discarded.

The testing environment is the soccer field which has standards specified by the Robocup legged robot league technical committee. The data used in the tests were collected using the field specifications of 2001 (Figure 3.10). The background is green with white strips showing the half line and penalty line, and covering walls are white. There are six artificial landmarks called beacons which are uniquely colored and have predefined position, size and colors to aid the robot to localize itself. There are two goal areas which are blue, and yellow. Recently, the size of the field was enlarged,

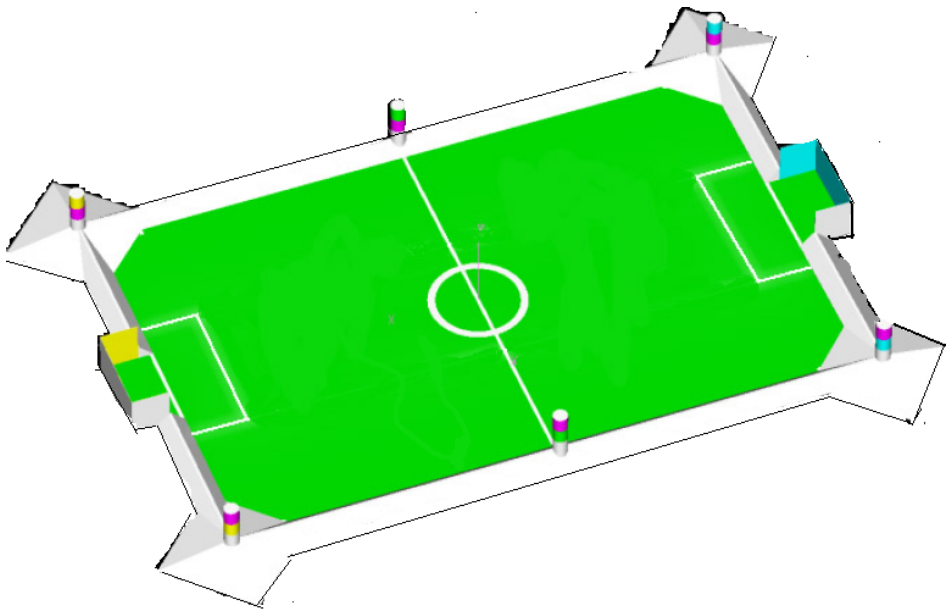

Figure 3.10. The old soccer field

which allows working with larger number of robots, but adds more uncertainty to the observations since now the landmarks are further away from the robots and it is harder to observe the distance to them accurately as the distance increases. In addition, two of the beacons, and the side walls are removed, to make the field look more like the real soccer fields (Figure 3.11). These changes make the problem more challenging.

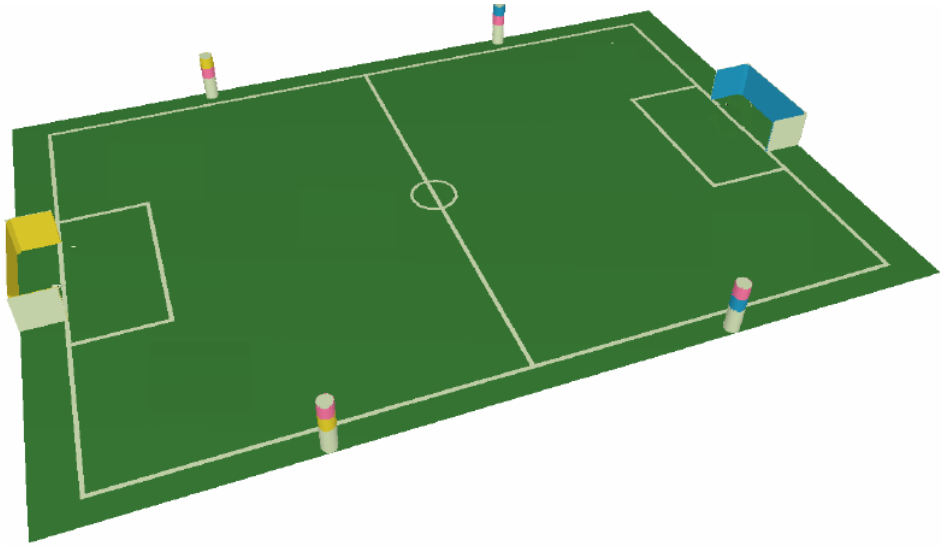

Figure 3.11. The new soccer field

3.4.2. Sony AIBO Robots

The single robot localization algorithms are tested on the ERS 210 type robot dog AIBO as seen in Figure 3.12 that is a commercial product of SONY (AIBO, 2005). These are quadruped legged robots which have an onboard CMOS camera with an angle of view 57.6 degrees in horizontal, and 47.8 degrees in vertical axis. The resolution of the camera is 176 x 144 pixels. The robot also has an IR sensor which work in the range 10-90 cm, and wireless LAN card that allows wireless communication between the robots. The robots used in the competition are dark gray with blue and red uniforms, which allow the other robots to detect them and decide whether they are opponents or teammates. Unfortunately identification of individual robots is not trivial, and therefore could not be implemented and is not currently used in localization.

Since 2004, the new product ERS 7 which has higher computational and physical power, is used by most of the soccer teams (Figure 3.13). The experiments on our multi robot algorithms were done on ERS 7 robots (Section 4.2).

Figure 3.12. SONY AIBO ERS 210

Figure 3.13. SONY AIBO ERS 7

3.4.3. Offline Tests

The experimental data for offline testing is provided from (Gutmann and Fox, 2002), where it it used for comparison of several well-known localization methods in literature. This now de facto standard set of test data is based on the records of the test runs of Sony's ERS 210 quadruped robot (AIBO) on the Robocup soccer field. The raw data are produced by running the manually manipulated robot on the field that is shown in Figure 3.14 (Gutmann and Fox, 2002) on a figure of eight like path for almost an hour, stopping the robot on several predefined points called markers, recording the perceptions of the landmarks and odometry readings during this run. The tests aim to analyze accuracy and robustness in case of noisy and sparse data, the time to recover from the kidnapping problem, and speed of the algorithms by measuring the time required for processing the frames.

In this work, the R-MCL algorithm is tested against the other outstanding methods whose details could be found in (Gutmann and Fox, 2002, Kristensen and Jensfelt, 2003). The results of the methods in (Gutmann and Fox, 2002) were provided by Gutmann. The algorithm is also tested against the S-Loc, and SRL* (Kaplan *et al*, 2006) algorithms which were implemented to be used in the same project in our laboratory. SRL* is shown with an asterisk to distinguish from the SRL implementation in (Gutmann and Fox, 2002), which had minor differences, and therefore different results in the tests.

Although the algorithms can make use of the goals and field lines to localize, the test data do not include this information. Additionally, although more than one beacon could be seen in one perception in the test data, since this is a very rare case in real games, it is not considered in the R-MCL tests. R-MCL is not specially trained and optimized for the test data, whereas the ones in (Gutmann and Fox, 2002) are, so this also affects the final success rates.

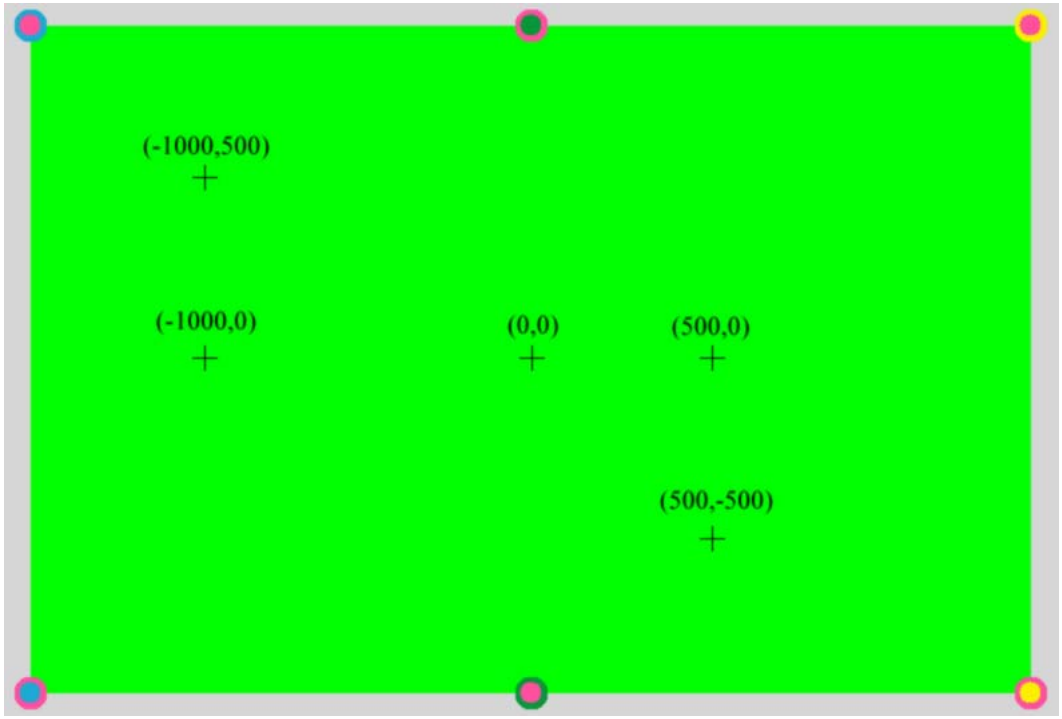

Figure 3.14. The soccer field of the test environment

3.4.3.1. Offline Testing Tool. For the offline experimental study, the offline testing tool developed in our laboratory is used. The user can choose one of the four tests

described in the next section, the data set, and one of the available methods which should be previously embedded to Cerberus Station code, which covers the offline codes of Cerberus Team. The user can also run the algorithm several times, and for different time durations in batch mode. The visual interface is shown in Figure 3.15.

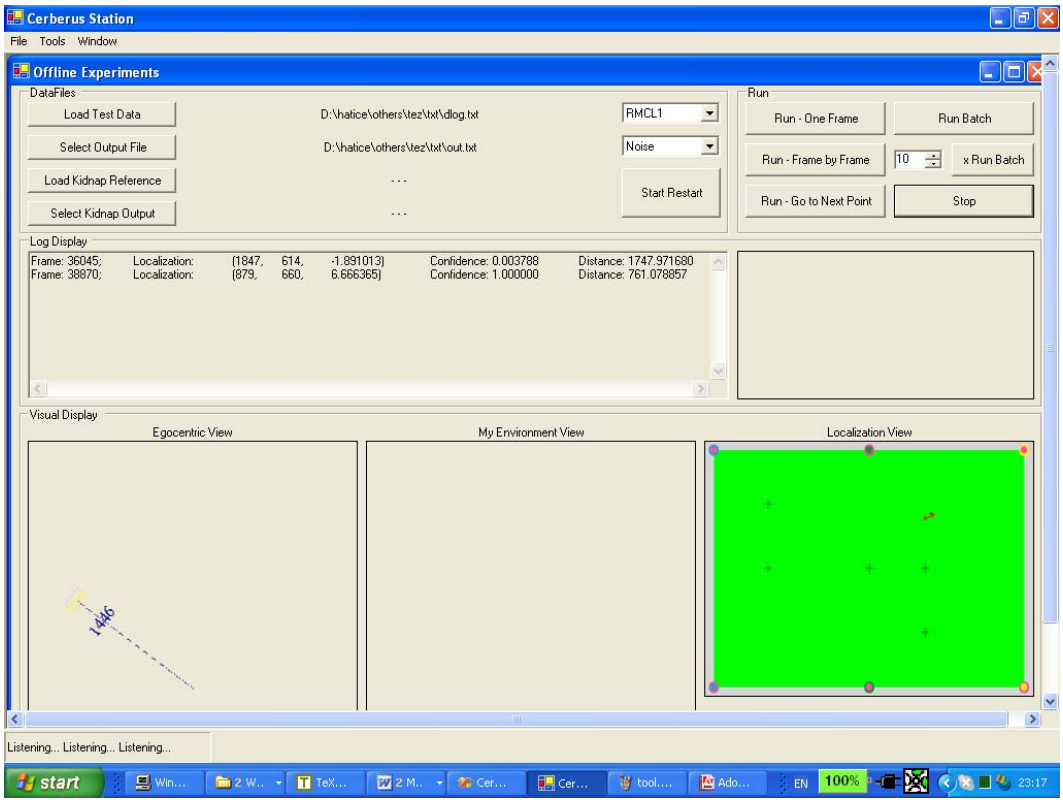

Figure 3.15. The offline testing tool

There are several display windows at the bottom of the visual interface, for displaying the distance error, certainty and the estimated positions of the robot at each predefined marker. The perceptions and estimated positions are also displayed visually. Besides there are several special purpose windows for special methods, like the grid window for RMCL, and the ME window for SLOC.

3.4.3.2. Noisy Data Test. In the noisy data tests, the percentage of noise level on the raw data is increased from 0 percent to 80 percent by 10 percent increments in every data set. The number of noisy samples is increased to assess the robustness and accuracy of observed methods in case of high noise levels.

In Figure 3.16, the error rates of the R-MCL, S-Loc, and SRL* are presented in

case of noisy data. The results of the previous versions of R-MCL can be found in (Köse *et al*, 2004, Köse and Akın, 2005, Köse *et al*, 2005). The comparison of R-MCL with other outstanding methods in the domain are presented in Figure 3.18. The error rates of the tests are calculated from the distance between the expected location of robot when it reaches a marker, and its exact location (the location of the marker). Note that, there are also unavoidable errors in the manual measurement of the exact locations of robot, for the evaluation of the test cases, due to experimental problems reported by the data providers.

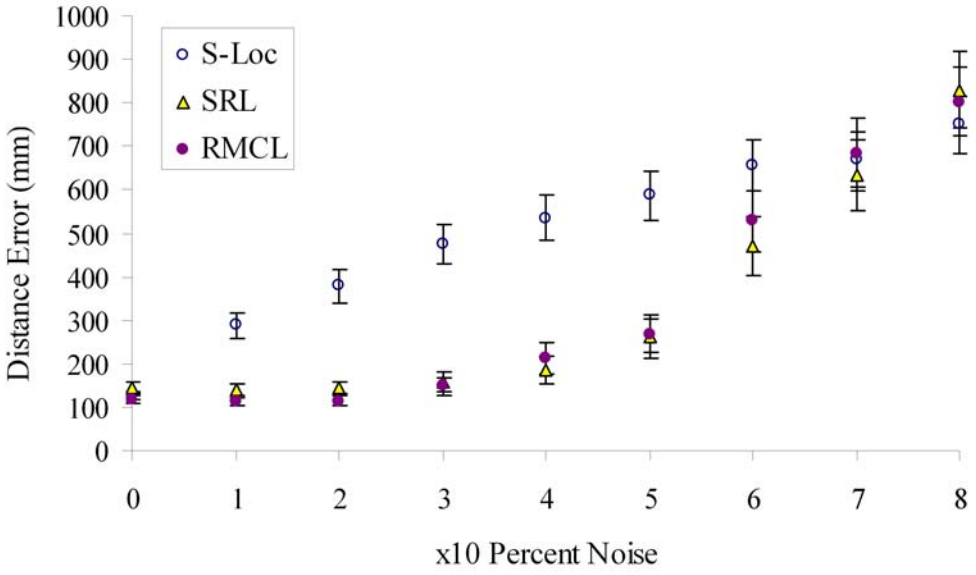

Figure 3.16. Results of the noisy data tests-1

In Figure 3.17 the error rates of the R-MCL, Geometric Localization (GEO), and Fuzzy R-MCL (FUZZY) are presented. R-MCL and Fuzzy R-MCL show almost similar behaviors, and outbeat the GEO method which is simple, fast but not robust to high noise, since it assumes perfect data.

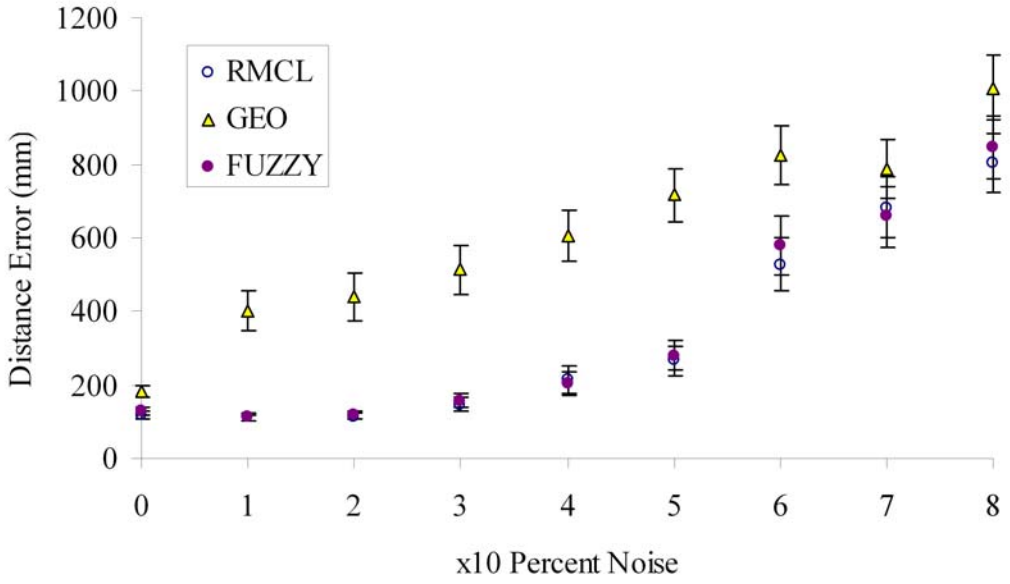

Figure 3.17. Results of the noisy data tests-2

The R-MCL algorithm shows a good overall performance. At the very high noise levels which are not realistic, its performance decreases, due to the ML module which is robust but coarse. As mentioned previously, the R-MCL algorithm is grid based. In the referenced works typically the cell size is chosen as 5 cm. However, in this work, it is taken as 25 cm for the R-MCL, to increase the speed. The triangulation method which is considered in case of observing two or more landmarks is also not used in the implementations. Using a smaller cell size and triangulation would decrease the error rate considerably, but the current case is more realistic. The bigger cell size is advantageous in fast recovery from kidnapping, and makes the algorithm quite fast by decreasing the number of cells drastically, but using the odometric data for big cells is useless. The odometric data are in terms of millimeters in every frame, and it is applied to cell centers, so unfortunately it is impossible to update the cell confidence correctly, or detect if the robot passes from one cell to the other by odometry. So odometric data are useful mostly in the MCL part of R-MCL. Unfortunately odometry is really vital especially in the sparse data case where the data are very rare.

In the way noisy data sets are prepared for the experimental study, only the false perception of the beacons is modeled, which constitutes only a small part of the noise problem and occurs infrequently (Gutmann and Fox, 2002). The main problem

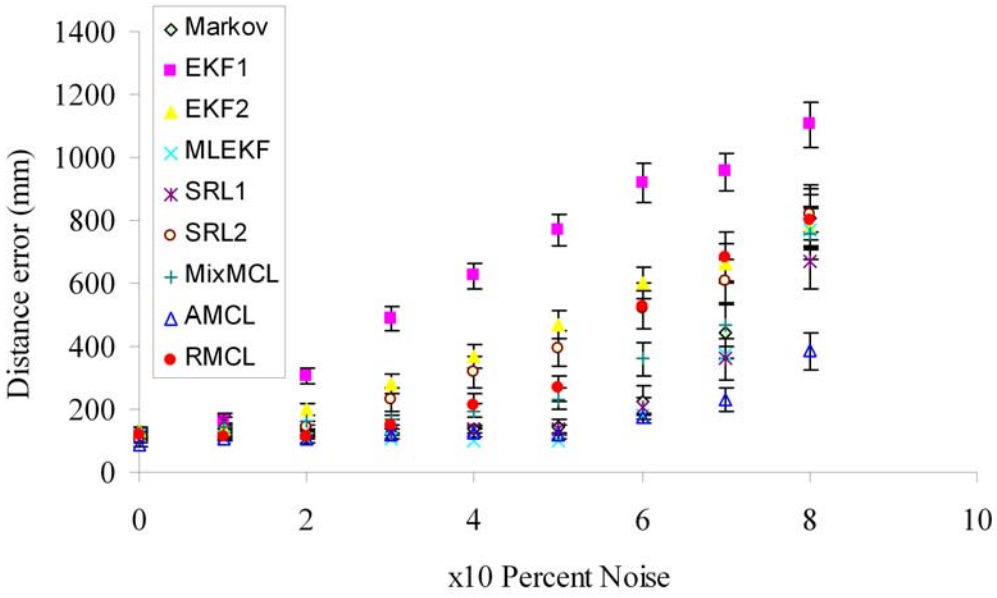

Figure 3.18. Results of the noisy data tests-3

with the perception observed in soccer games was actually the noise in the distance estimation, which can be better tested in real time tests discussed in Subsection 3.4.4.

Different parameter sets for R-MCL were also tested to find the optimum choice. First of all, different th_{ML} values were compared. The results of the noise tests for different th_{ML} values such as 10, 15 and 20 are compared. The value of 15 gives the best result as seen in Figure 3.19.

In these implementations, static samples are used to cover the field better with smaller number of samples. In Figure 3.20, random and static samples are compared in terms of noise. Random samples are also drawn from the selected cells. Static samples give slightly better results.

3.4.3.3. Sparse Data Test. In sparse data tests, samples are deleted from the original raw data, in a predefined sequence, (beyond the robot's awareness). The sparsity increases from 1/1 to 1/256, by 2^{-n} of sparsity increases in every data set. As the frequency increases, the behavior of the selected methods is observed.

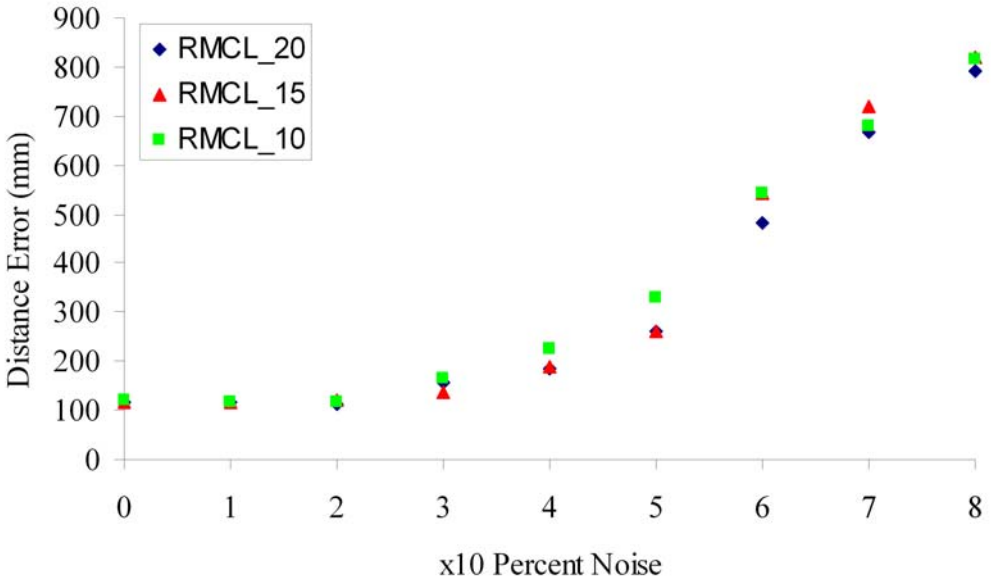

Figure 3.19. Results of the noise tests for different th_{ML}

The error rates of R-MCL, S-Loc, and SRL* are presented in case of sparse data (Figure 3.21) It should also be noted that S-Loc is using the ME output instead of using the perception data directly, which is an important advantage especially against sparsity. Also note that, R-MCL showed better performance than SRL* although it uses fewer samples than SRL* even in the worst case. The comparison of R-MCL with other outstanding methods in the domain are presented in Figure 3.22. It performs well up to an acceptable sparsity level. In the very high levels of sparsity which can not be normally encountered in real games and experiments, its performance is not as good as others. This is because it is not using a memory and uses small amount of samples.

In Figure 3.23 the error rates of the R-MCL, Geometric Localization, and Fuzzy R-MCL are presented. R-MCL outperforms others. GEO shows a satisfactory behavior although it does not use any uncertainty model, or samples. Fuzzy R-MCL is beaten by the others.

In Figure 3.24, the results of sparsity tests for different th_{ML} values such as 10, 15 and 20 are compared. The value of 15 gives the best result, as it does in the noise tests.

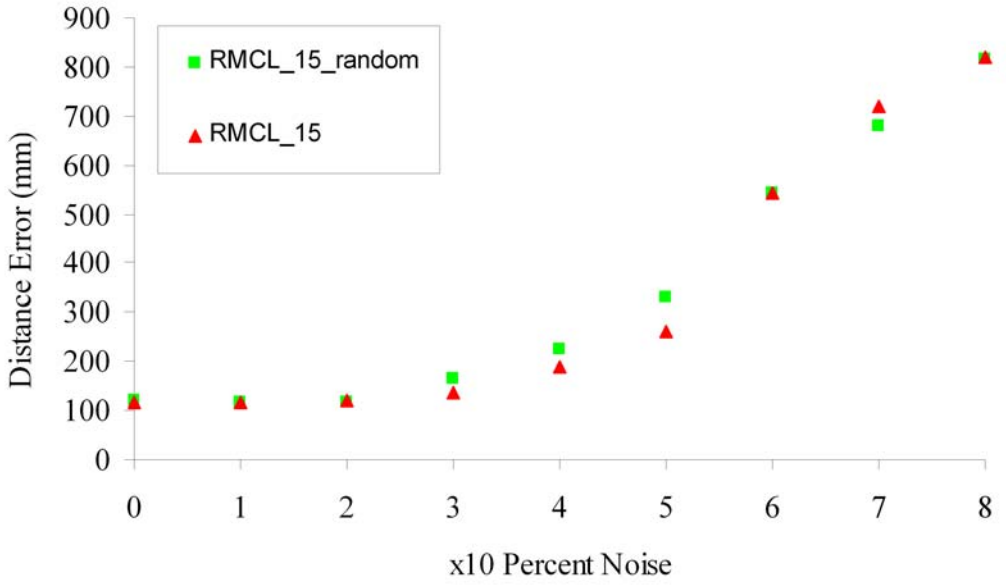

Figure 3.20. Results of the noise tests for comparing the static vs. random samples

In Figure 3.25, random and static samples are compared in terms of sparsity tests. Random samples are also drawn from the selected cells. Static samples give slightly better results, as observed in the noise tests.

Adaptive thresholds, adaptive number of samples and grid sizes, for different levels of noise and sparsity could be used to increase success. Notice that the algorithm has an error rate which is not greater than the outstanding methods in the domain up to a high error and sparsity level, which are too high to be realistic, indeed. So the error rate of the method is acceptable for real life applications.

3.4.3.4. Recovery from Kidnapping Test. In the third experiment, the ability of the methods to solve the *kidnapped robot* problem is investigated. In this problem, the robot is displaced without being aware. Since, its beliefs are wrong, it should reset and localize itself from scratch. In this experiment, the average time the methods need for re-localizing the robot after it has been kidnapped is computed over 22 kidnapping tests. The results are shown in Figure 3.26 where the values are in seconds. According to the experimental results, R-MCL recovers from kidnapping faster with the help of its ML part.

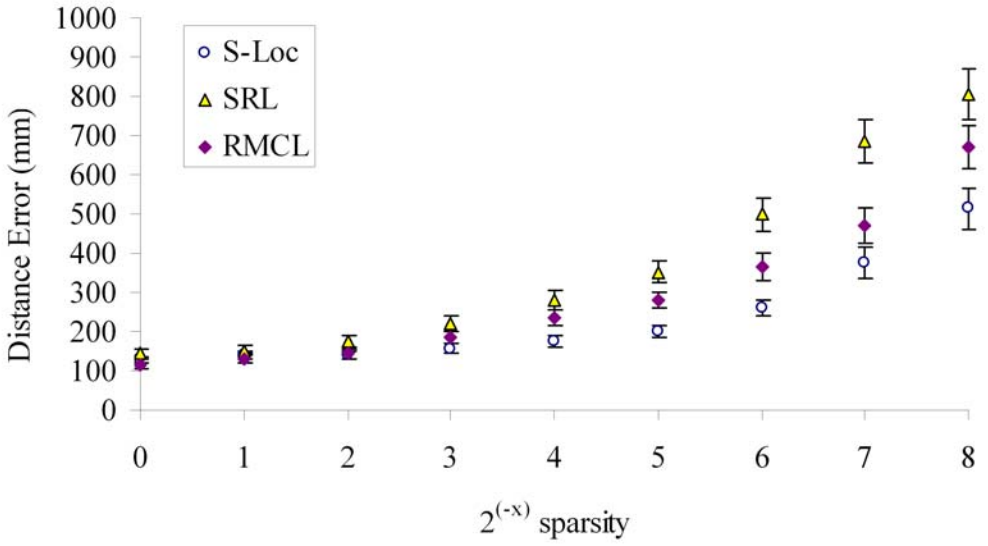

Figure 3.21. Results of the sparsity tests-1

In Figure 3.27 the kidnapping test results of the R-MCL, Geometric Localization, and Fuzzy R-MCL are presented. R-MCL and Fuzzy R-MCL show similar behaviors and a very high success rate. GEO has a high error rate and, is beaten by the others.

3.4.3.5. Speed Test. In this experiment, which is carried on an ERS 210 robot, the average processing time of the methods are tested. In order to have a fair comparison, the only active process in the memory is the localization process. The average processing times and the number of processed frames per second are calculated for each method over the complete raw data set that contains 51523 frames ten times. For SRL and R-MCL, the processing time interval is the average of each frame's time intervals for the motion and vision updates for localization module; whereas for S-Loc, the time intervals include motion and vision updates for the ME module as well. As shown in Table 3.1, where results are in microseconds, the total processing time for S-Loc and ME is much less than both SRL and R-MCL whereas R-MCL is more than twice faster than SRL. R-MCL uses big grid sizes and picks up samples only when the number of good grids is below a threshold, so the total number of processed grids and samples are less than that of SRL even in the worst case. This increases the computation speed and decreases the memory needed to hold the samples, but accuracy is also lower.

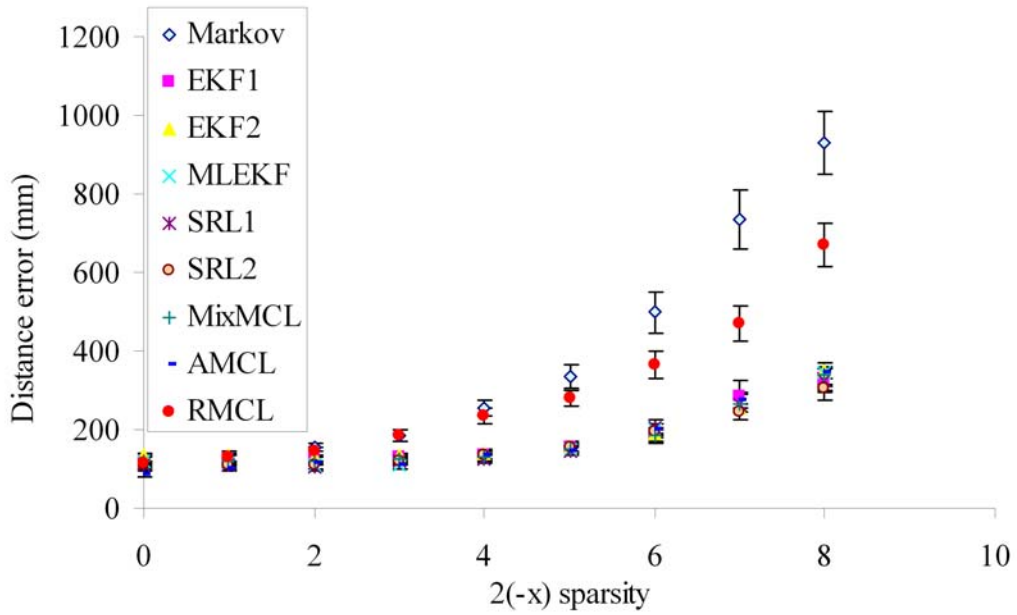

Figure 3.22. Results of the sparsity tests-3

Table 3.1. Results of the speed test on real robot ($\mu sec.$)

	S-Loc	**SRL***	**R-MCL**
Processing Time	858	10484	4942

The experiment is also tested on ERS 7 robots. These robots have higher computational power than ERS 210 robots, as stated before. The results are presented in Table 3.2, where results are in microseconds, again the total processing time for S-Loc and ME is much less than both SRL and R-MCL where as R-MCL is more than twice faster than SRL.

Table 3.2. Results of the speed test on real robot ($\mu sec.$)

	S-Loc	**SRL***	**R-MCL**
Processing Time	155	1839	923

In Table 3.3 the results on PC for R-MCL, Geometric Localization, and Fuzzy R-MCL are presented. Fuzzy R-MCL shows better performance than R-MCL, and both have a good success rate. GEO has a very good timing, since it uses a very simple working schema, and do not use any samples or grid cells at all.

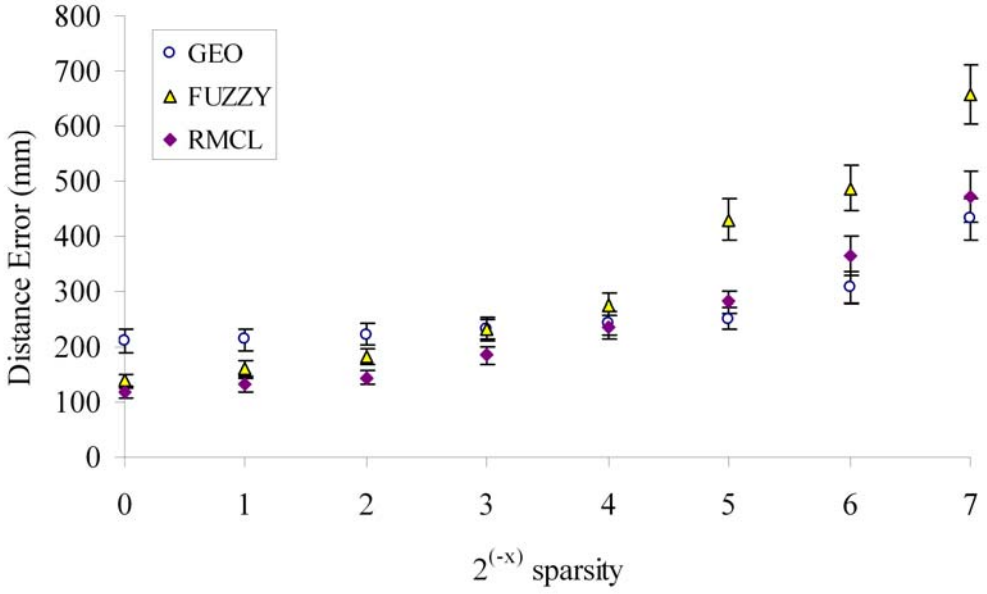

Figure 3.23. Results of the sparsity tests-2

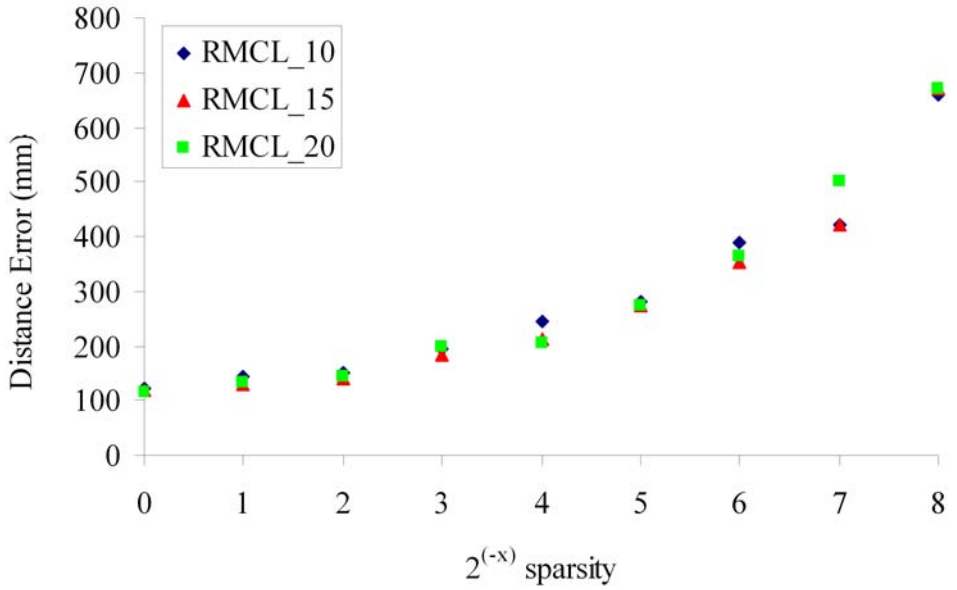

Figure 3.24. Results of the sparsity tests for different th_{ML}

Table 3.3. Results of the speed test on PC ($\mu sec.$)

	GEO	**FUZZY**	**R-MCL**
Processing Time	181	284	383

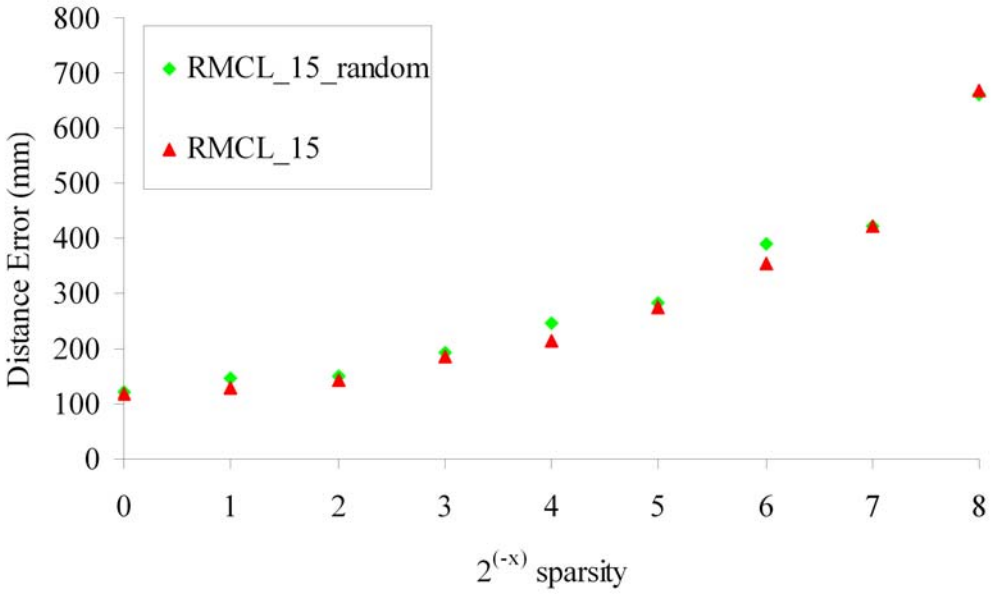

Figure 3.25. Results of the sparsity tests for comparing the static vs. random samples

3.4.4. Real Time Tests

In order to observe the performance of the methods in the real environment several real-time tests were conducted. The field used in these tests is the new field which has four beacons, and is enlarged by almost 1.5 times the old field used in the offline tests. The white walls around the field were also removed, so that the robot could detect any object in the lab around the field and might be confused. Besides the color table of the robot is an older one which was trained under different lighting conditions, and the lighting conditions of the test field varied due to variations in day light which could not be avoided. The scenario of the tests was taken from a localization challenge of Robocup games. There are six markers on the field, four of them placed nearby the beacons and two on the mid-field line, and the robot should visit all of these markers in a predefined sequence. In Figure 3.28, the markers are represented by light colored rectangles on the field (Although they are only points in the field their sizes are exaggerated to be seen clearly in the picture). Each time the robot is started from the center of the field. Whenever its distance to the current marker in the visiting list is below a threshold (10 cm in the current tests), and it is confident enough, the robot stops and wags its tail. During the tests, while the robot is following its path, it becomes kidnapped many times when it goes out of the field and is placed on the

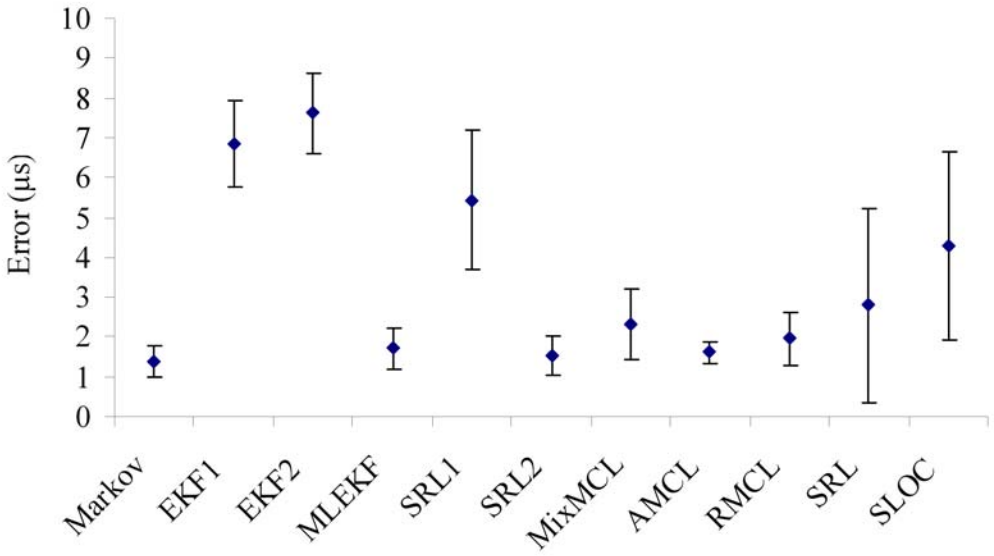

Figure 3.26. Results of the kidnapping tests-1

nearest position in the field. It was also kidnapped and placed on the opposite side of the field many times on purpose to measure its robustness against kidnapping. These facts besides problems due to the manual measurement and data collection increased the overall error unavoidably.

Each test is performed ten times and the average and standard deviation of the error in distance is calculated. Since the robot stops for every marker and is started manually again, it was not suitable to measure the time. So only the error in distance is measured in these tests.

We ran the four algorithms S-Loc, SRL* and two versions of R-MCL, according to this scenario. All of the parameters of R-MCL1 and R-MCL2 are the same with the offline tests, except th_{ML} which is 15 cm in the offline tests and R-MCL1, but 30 in R-MCL2. Since the field is larger, the field is represented with more cells now, so it is logical to increase the th_{ML} to improve the success. This also increases the number of samples used in the MCL part, but it is still in the acceptable range. In the tests, SRL* could not converge to the points in the limited time duration so its results are not taken into consideration. In these tests, R-MCL outperforms S-Loc and another version of itself, as seen in Figure 3.29.

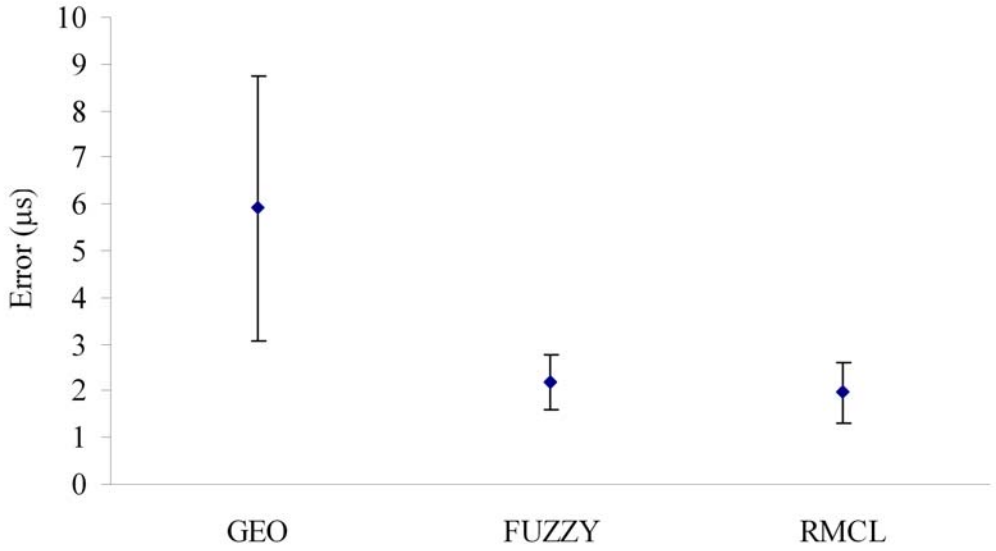

Figure 3.27. Results of the kidnapping tests-2

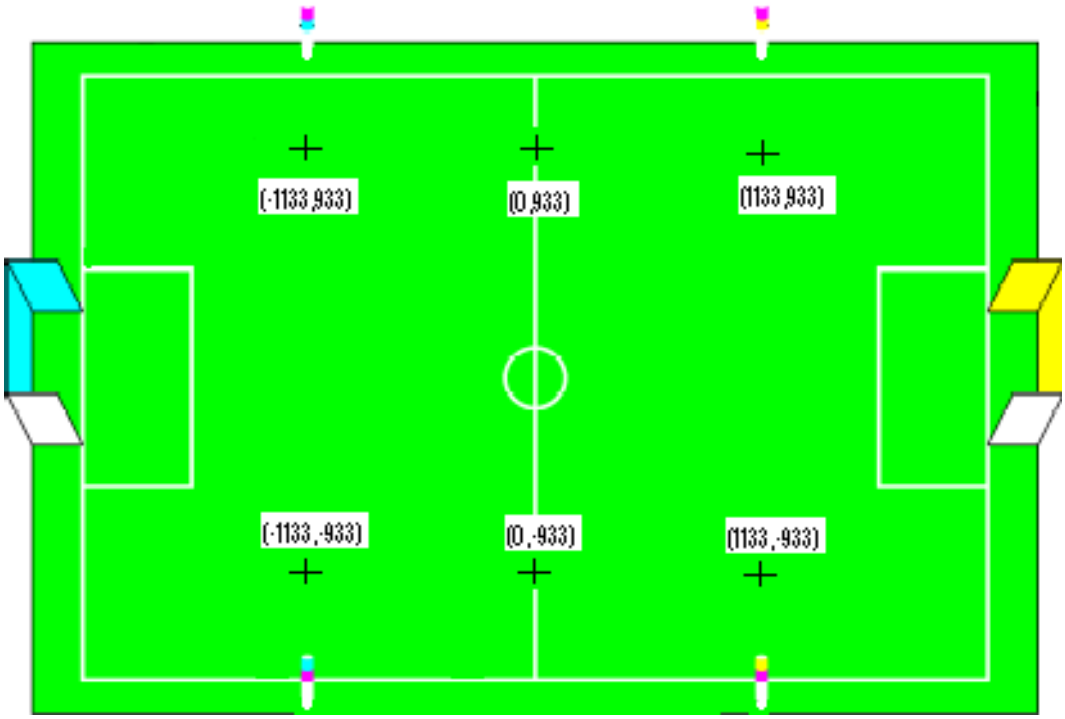

Figure 3.28. Real time test field with markers

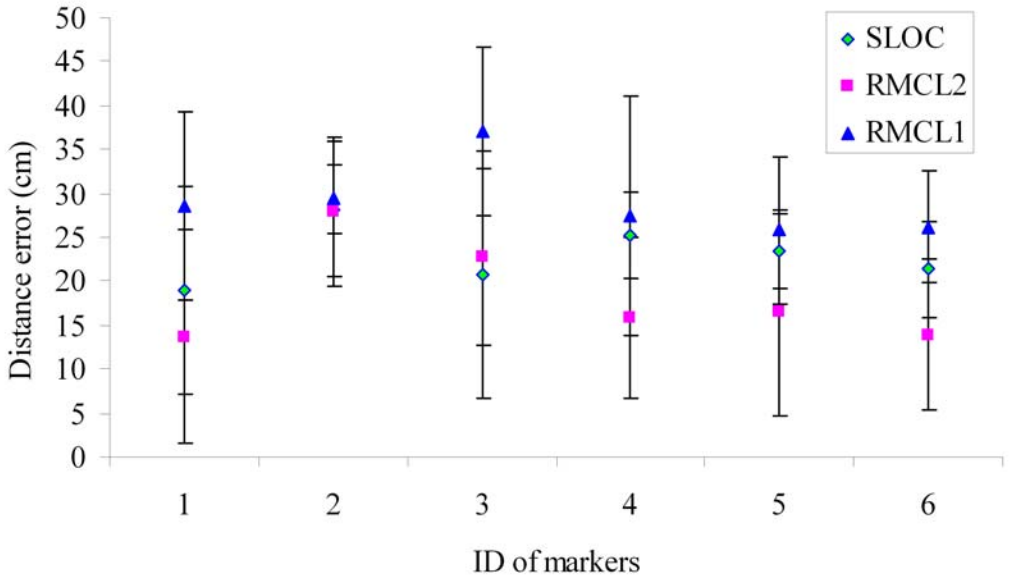

Figure 3.29. Results of the real time tests with six markers

4. COLLABORATIVE REVERSE MONTE CARLO LOCALIZATION

Multi-robot localization is a newly developing area in the robotics domain. In multi-robot localization, the aim is to decrease the localization error by using the shared perception and estimations between the team members. In this work a novel multi-robot localization method is proposed. Collaborative Reverse Monte Carlo Localization (CR-MCL) is a collaborative method based on R-MCL method. This method is tested both in the simulated environment and on real robots, and the results are presented in this chapter.

4.1. Collaborative Reverse Monte Carlo Localization Method

The proposed collaborative method is based on the R-MCL self-localization method described in Section 3.2. Each robot in the team is assumed to be capable of self-localization using R-MCL by means of its sensors and actuators. But the accuracy and speed of this self pose estimation may vary due to the environmental and intra-robot conditions. In R-MCL, each robot has its local grid cells to estimate its own position. When the certainty of the robot about its location increases, it uses a sample set based on this grid set, initially. Since it is based on R-MCL, also in CR-MCL, when two or more robots encounter each other, they represent the position of the robots they detected in terms of grid cells, too. Each robot m that detects another robot n, updates its local grid cells using the relative distance and orientation it observes between itself and n, to produce the new shared grid cells estimating the position of n. For every possible location l (in our work grid cell centers),$Bel_{nm}(L = l)$, the belief of robot m in robot n's being in location l is calculated (Equation 4.1). The calculation uses the following

- the probability of m observing n in the location l, while itself being in the possible locations l', and

- the detection information r_m, which presents the relative information between m and n observed by n, and sent to n

n then integrates these new grid cell beliefs coming from all m into its own grid set as given in Equation 4.2 based on (Fox *et al*, 2000). Here the prior probabilities are calculated with similar models that are used in ML method of the R-MCL (Section 3.2).

$$Bel_{nm}(L = l) \leftarrow \sum_{for\ each\ l'} P(L_n = l | L_m = l', r_m)\ Bel_m(L = l') \tag{4.1}$$

$$Bel_n(L = l) \leftarrow Bel_n(L = l) \sum_{for\ each\ m} Bel_{nm}(L = l) \tag{4.2}$$

To describe the algorithm better, we present a two robot scenario on the test field. In Figures 4.1 and 4.2 the robot observes two beacons simultaneously, and the grid cell probabilities due to these observations are presented. In the figures light colored cells are the ones with the highest probability. The rest of the cells have negligible small probabilities and are represented with dark colors. In Figure 4.3 the resultant grid cells to be used by the robot to localize itself, are presented. These are formed by the integration of the first two sets based on the observations. Here the robot uses R-MCL to self-localize itself.

In the two robot version of this scenario, in Figure 4.1, $robot_1$ sees one beacon. In Figure 4.4 $robot_2$ also sees one beacon. Both robots form local grid sets representing their position estimations using these observations. Then $robot_2$ sees $robot_1$. Using its local grid set, and the relative distance and orientation it observed between itself and $robot_1$, $robot_2$ produces a new shared grid set representing its estimation on $robot_1$'s position (Figure 4.5). It sends these cells to $robot_1$. In Figure 4.6 $robot_1$ integrates these to its local grid set, to localize itself as if it sees two beacons as in Figure 4.3.

The CR-MCL algorithm is presented in Figure 4.7. This is a modified version of the R-MCL algorithm. Whenever a robot observes another robot it produces a shared

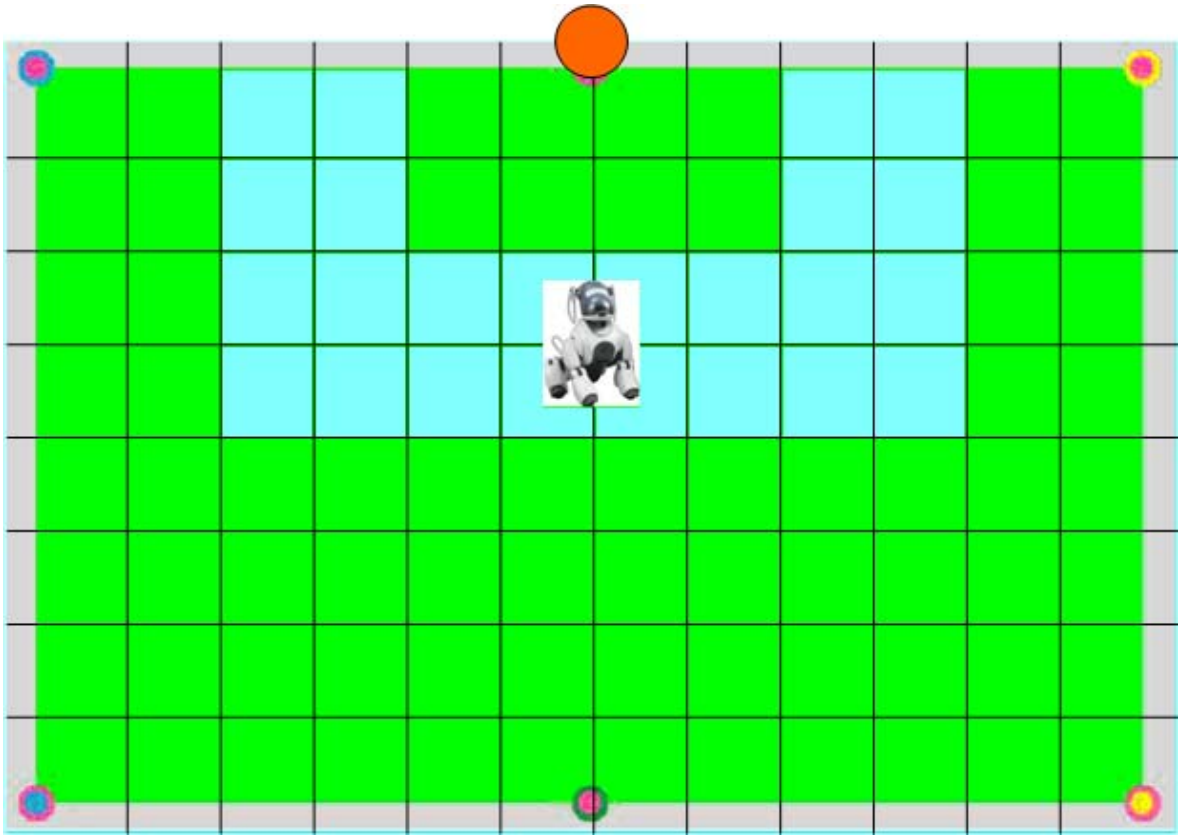

Figure 4.1. $robot_1$ sees one beacon

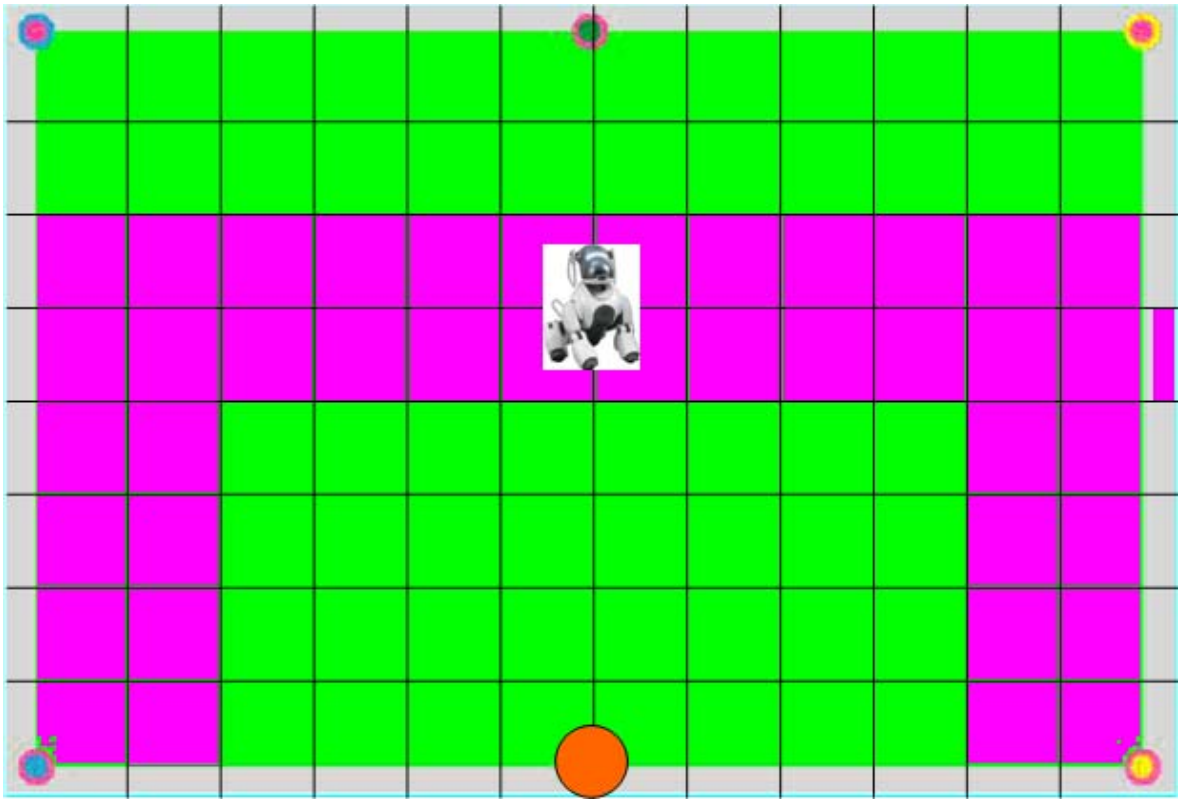

Figure 4.2. $robot_1$ sees a second beacon

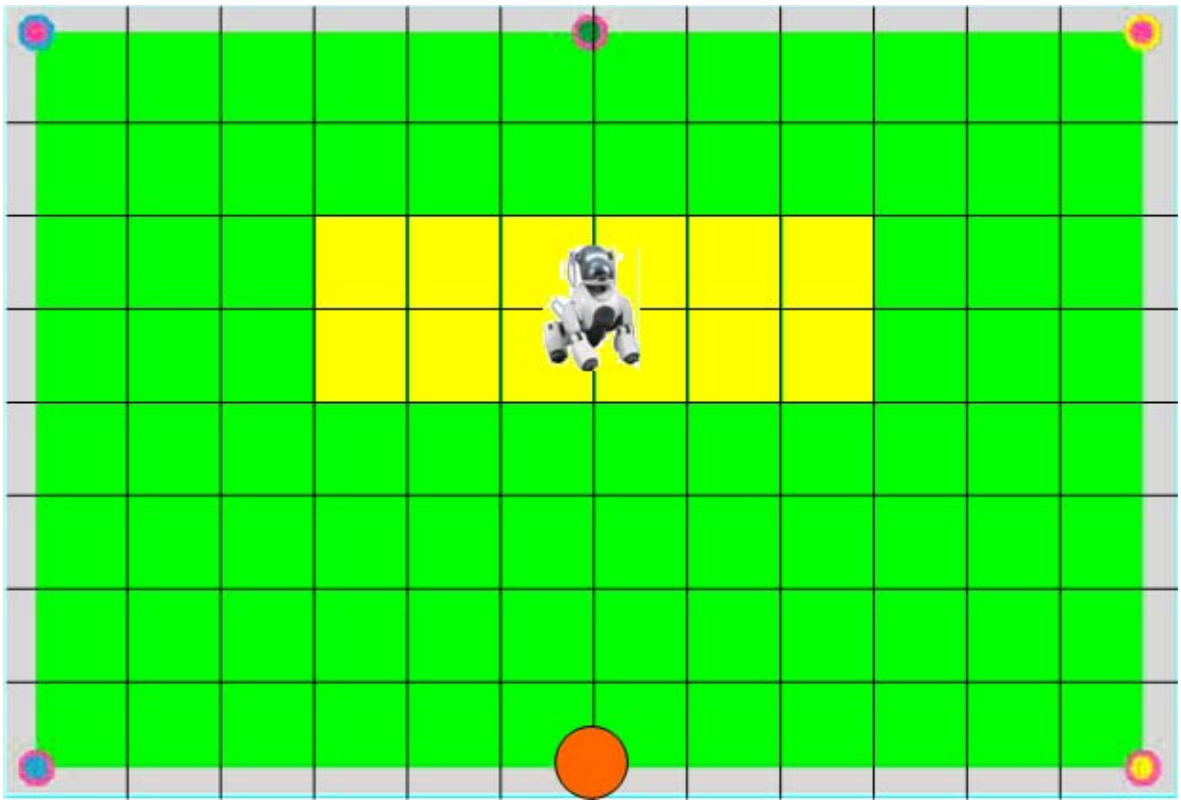

Figure 4.3. $robot_1$ localizes itself with using two beacons

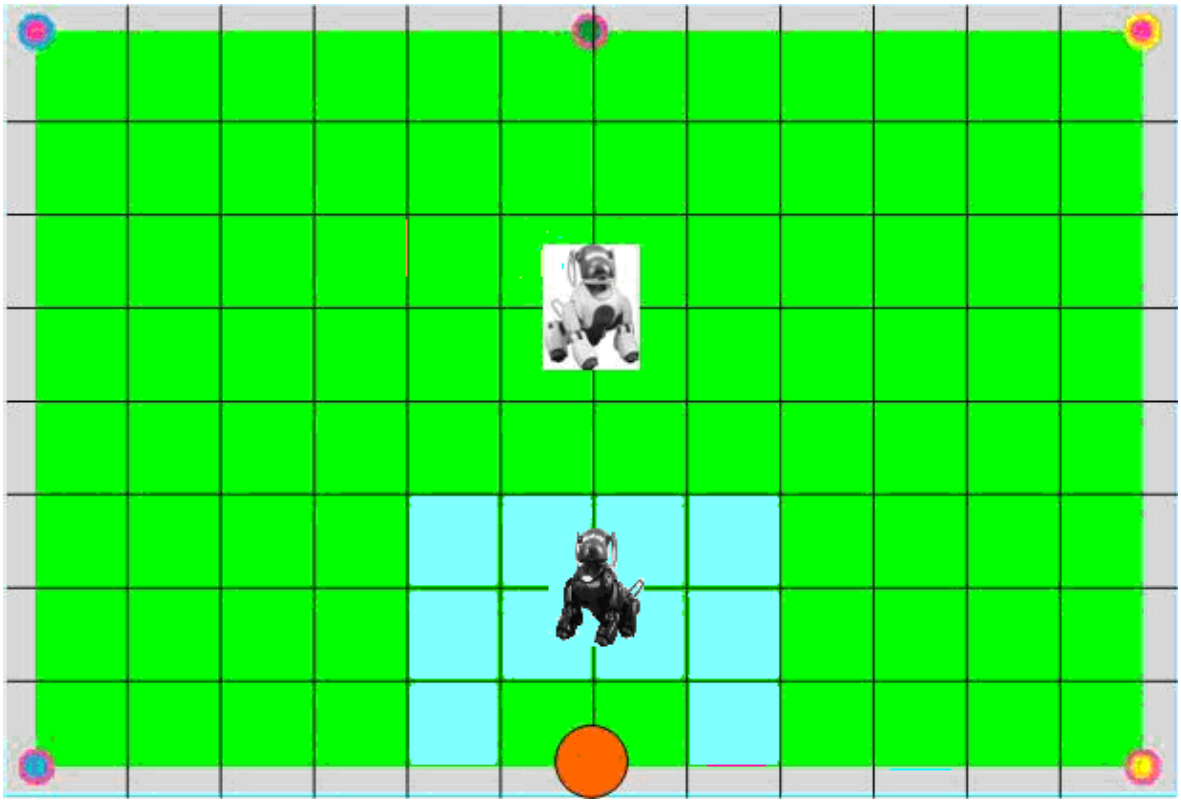

Figure 4.4. $robot_2$ sees one beacon

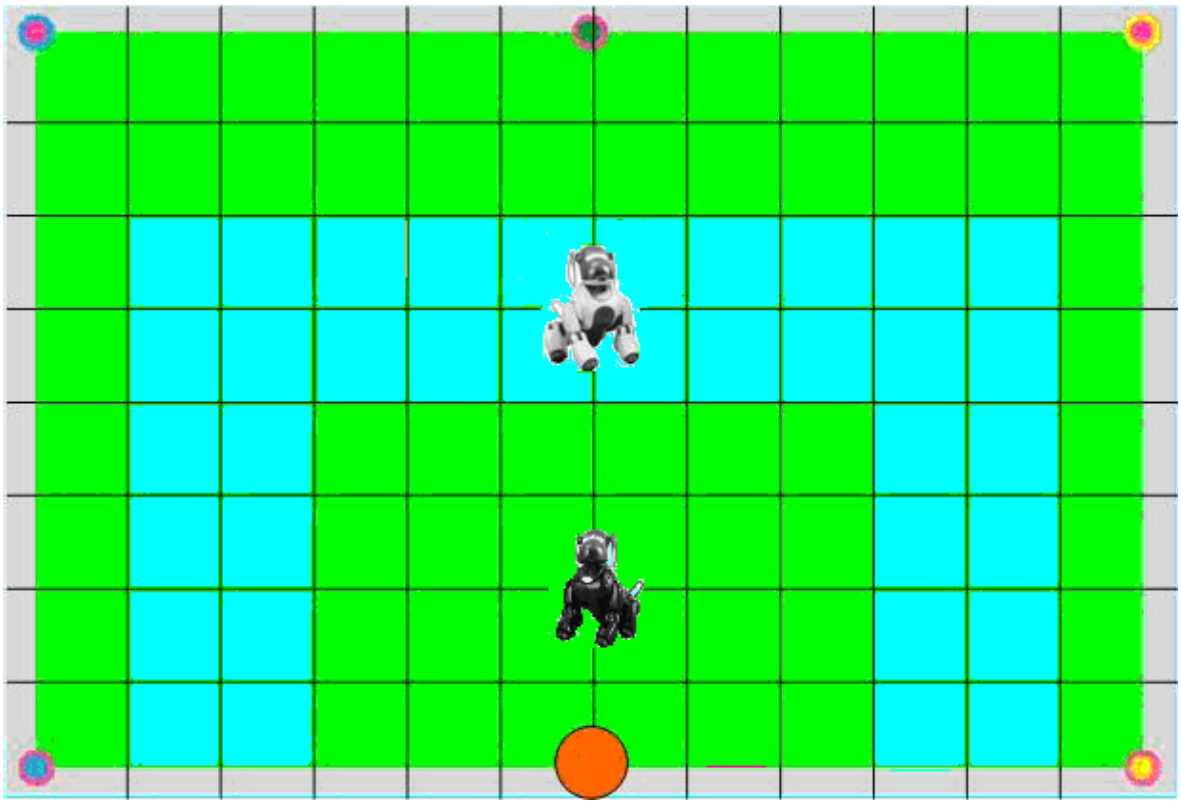

Figure 4.5. $robot_2$ estimates $robot_1$'s location

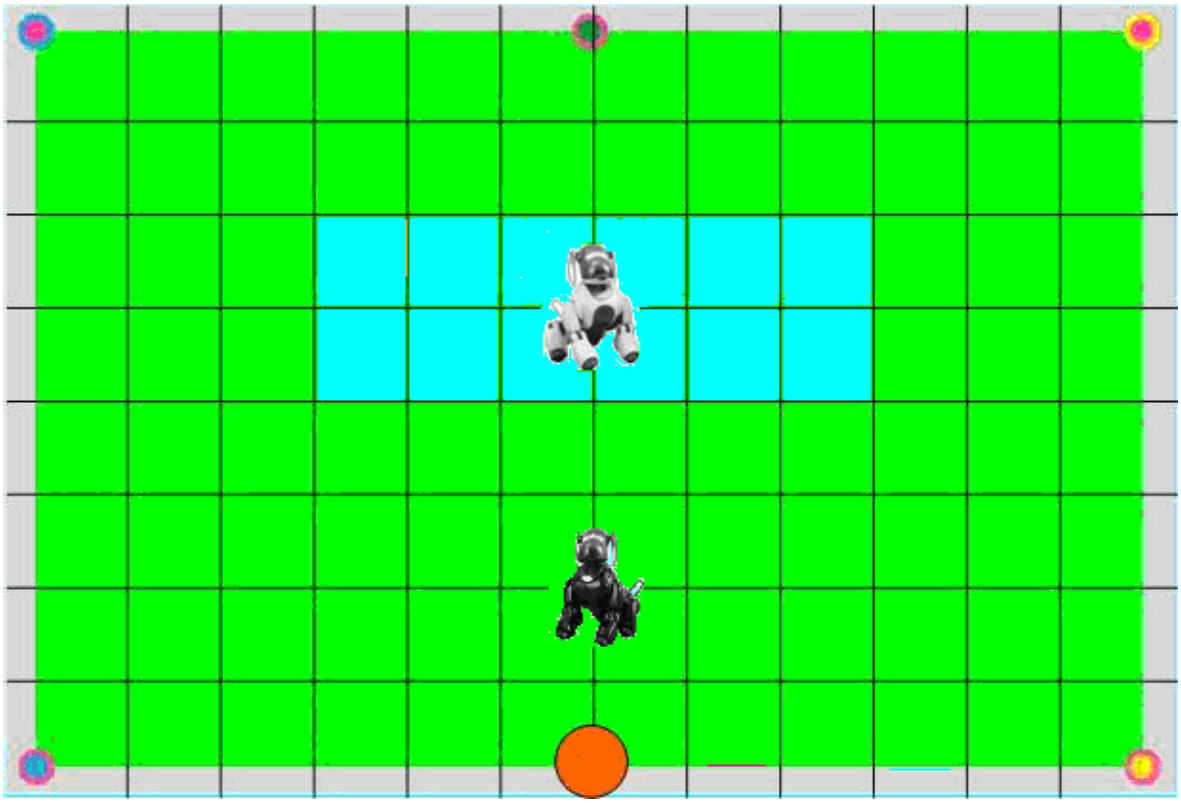

Figure 4.6. $robot_1$ localizes itself integrating the shared and the observed information

grid set. Observations are represented by grid cells instead of samples since observations are not very accurate. This approach is more robust, and has less computational and communication cost, since the same information is represented by fewer points. If the robot is very accurate about its place and using samples to localize itself, it uses the most recently updated grid set it used to produce its shared grid set. If the observed robot is very accurate about its location (it is in MCL mode) it does not use the shared information since it is not as accurate as its current estimations.

procedure $CR-MCL(local\_max\_grid\_array, shared\_grid\_array, bool\_ML)$

```
1: if bool_ML==TRUE then
2:    ML_update
3:    integrate(local_max_grid_array, shared_grid_array)
4:    if ML_number_of_grid_cells_in_max_grid_array < Th_ML then
5:       MCL_init(ML_samples)
6:       bool_ML=FALSE
7:    end if
8: else
9:    MCL_update
10:   MCL_init(ML_samples)
11:   if MCL_lost()==TRUE then
12:      ML_reset()
13:      bool_ML=TRUE
14:   end if
15: end if
16: update(shared_grid_array)
17: send(shared_grid_array)
```

Figure 4.7. The CR-MCL Algorithm

Using the shared information, we aim to increase the accuracy of pose estimations of the whole team. Unlike other works in the same domain, if the uncertainty of observation or the uncertainty of the detector robot about its own location is too high, then this observation is not used by the detected robot, to filter out useless shared data which will increase its error and uncertainty. By sharing only information with relatively high certainty, we expect to improve success, decrease computational and communication cost, and increase the benefit taken from the collaboration. If the detected robot is too confident about its location, it does not accept shared data, since shared information is not very accurate, and communication is costly. In addition,

not the whole grid set, but only the chosen grid cells with the highest certainties are transferred as shared grids. This is also expected to decrease transfer and calculation costs. e.g. if the whole test field is represented by 100 grid cells, and if a robot with considerable certainty uses only 20 of them with the highest certainties, then the number of new shared grid cells will be at most 20 whose calculation and transfer will be considerably lower than 100 cells.

4.2. Tests and Results for Multi-Robot Localization

To verify the predictions, the method was tested on several AIBO ERS 7 robots both using a simulator, and in the real field, which is the official field of Sony 4-legged League (Sony Four-Legged Robot League, 2005). The field is the new field which is larger and has only four beacons and no white walls which makes it more challenging. Detailed information about the simulator, field, and robots is given in Section 3.4.3. Some of the selected tests which cover the problem best are presented here. The input data used for localization is the same as the single robot case, such as the relative distance and orientation of robots to the beacons, observed using their onboard cameras. In the tests, the observed robot is movable and additionally uses odometry for localization. The observer robots are kept stationary, for simplicity.

In the simulator tests, there is no additional noise in the input data, but the robots suffer from lack of adequate information for localization. The observed robot moves 20 steps in the half field. Two stationary observers can observe the beacons and the observed robot at each step, localize themselves and send the estimated position of the moving robot. It uses shared data and its own observations to localize itself. The robots can observe at most two beacons, as stated in Table 4.1. The test results in Figure 4.8 show that the best results are obtained when the moving robot can observe two beacons. The tests where it can use shared data containing two beacons show similar success. In the tests where data from one beacon is observed or shared, the success rate is almost as half as the two beacon case. In the worst case the robot is blind (can not see or share anything) and can only use dead reckoning from its odometric data.

Table 4.1. Test descriptions

Test ID	Test Description	
	Detected Robot	**Observer(s)**
T1	Robot can see 1 beacon	1 observer can see 1 beacon
T2	Robot can not see	2 observers, each can see 1 beacon
T3	Robot can not see	1 observer, can see 1 beacon
T4	Robot can not see	1 observer, can see 2 beacons
T5	Robot can not see	no observers
T6	Robot can see 1 beacon	no observers
T7	Robot can see 2 beacons	no observers

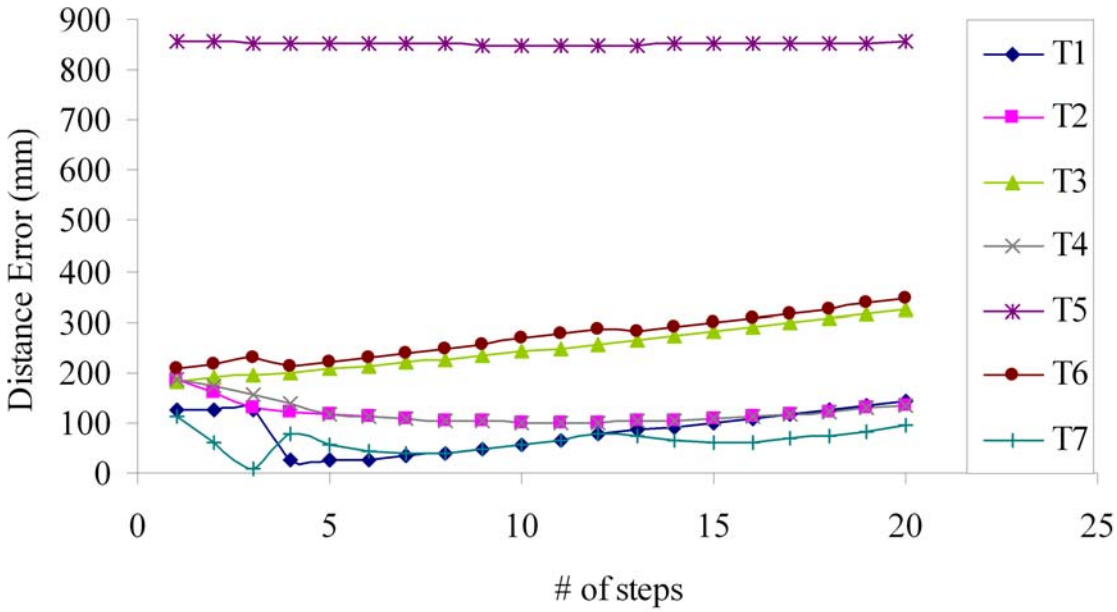

Figure 4.8. Test results in the simulator

In the real field tests, four stationary observer robots are placed equidistant to each other, and one meter away from the upper half of the field, dividing the whole field evenly. The observed dog is placed on four points on a path equidistant to observers and the beacons, as in the simulator tests (Represented by '+' on the field in Figure 4.11). Unlike them, the observed robot is always blind. There is high vision error due to imperfect lighting, and other irrelevant objects in the field and its surroundings, e.g. posters on the walls, besides the actual positions of the robots on the field are measured manually, which inevitably increases localization error. As seen in Figure 4.9, the first four results are the estimated positions of the observed robot calculated by robot $R0, R1, R2$, and $R3$, without collaboration. The last one is the result of the collaborative R-MCL. It is significantly better than the rest, including the best ($R3$), and average of the non-collaborative estimations.

In the second set of tests, the observers are placed 1.5 meters away from the upper part of the field. The location of the observed robot is the same with the first set of tests. In this part of the field the lighting conditions were better, and the robots can observe the beacons better so the results were better then the first set of tests. However, as expected the relative ratio between the results were almost same with the first set of tests, and CR-MCL has the best result again (Fig. 4.10). These tests prove that regardless of the distance between the observers, and the observed robot, CR-MCL method works successfully, and outperforms non-collaborating agents.

The real world tests are also repeated in the simulator under the same conditions but without additional noise. So the perception and action information coming to the robots are perfect. In Figure 4.12, the observed robot is placed 50 cm away from the top of the field and the observers are placed 100 cm away from the top of the field. In this test, as in the real field tests, CR-MCL performs better than both the best and average values.

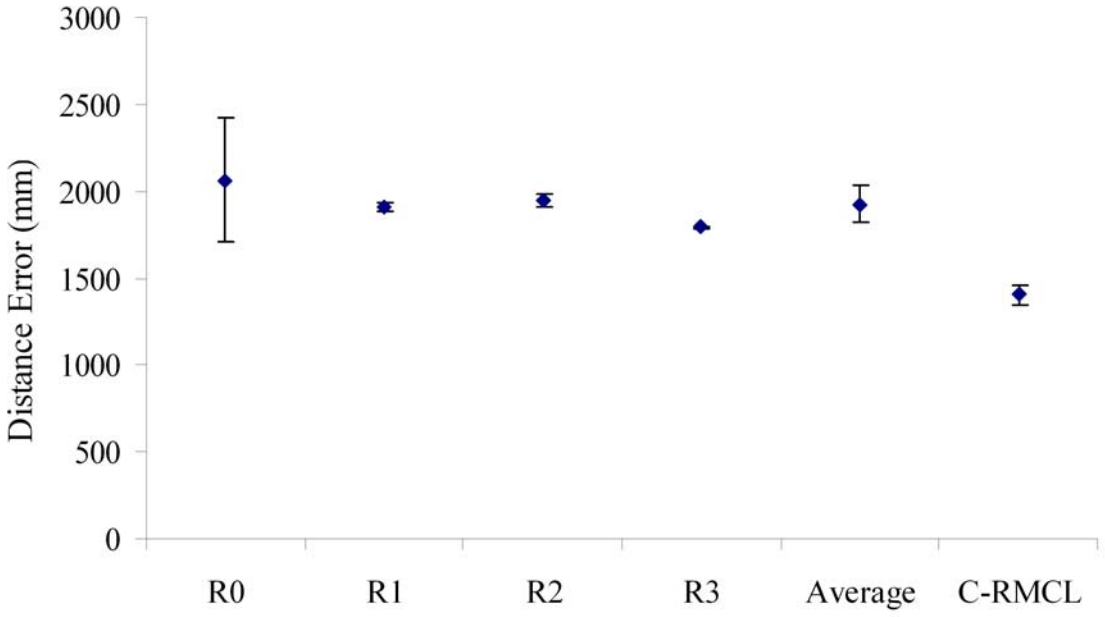

Figure 4.9. First set of test results in the real field

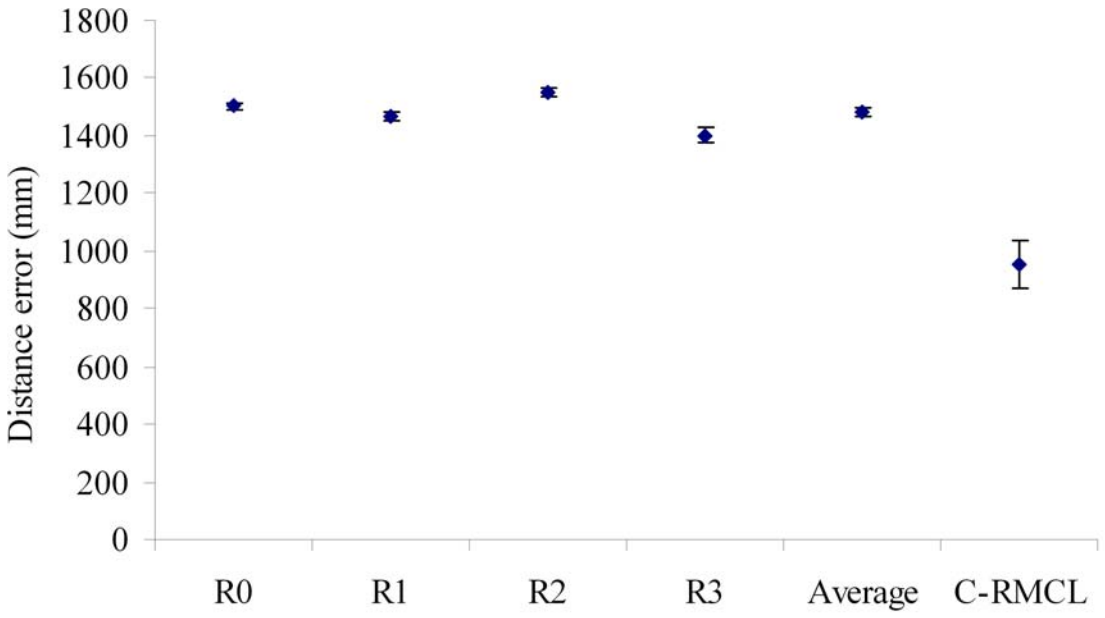

Figure 4.10. Second set of test results in the real field

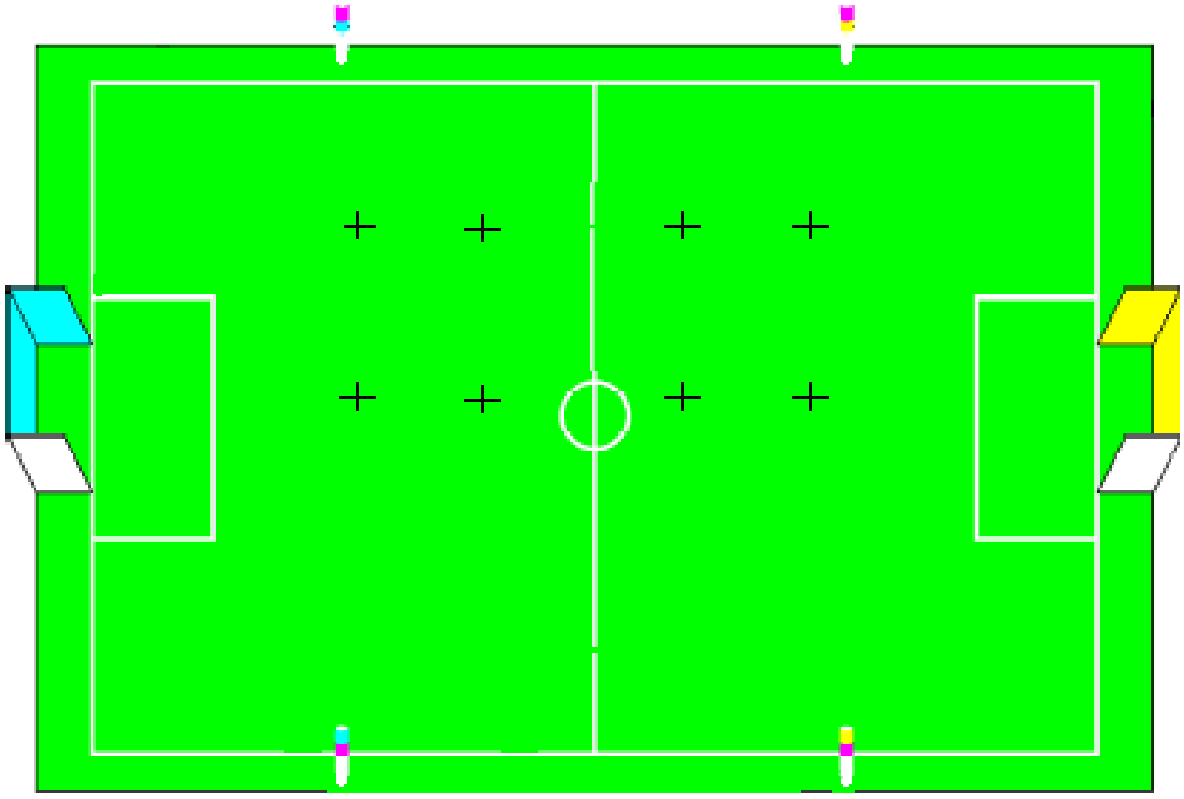

Figure 4.11. The real test field

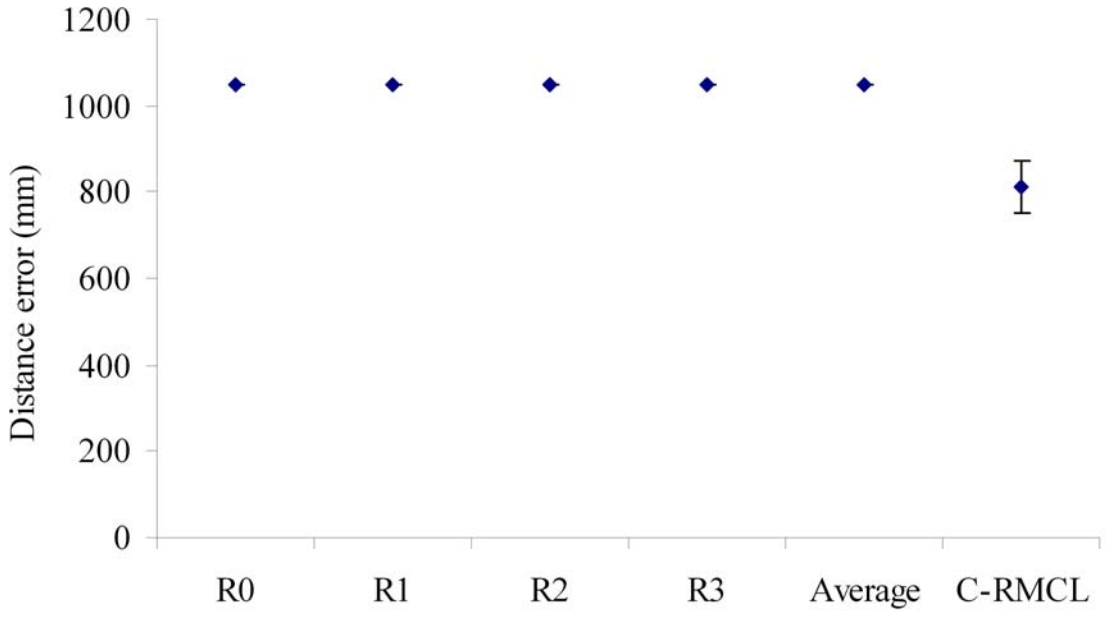

Figure 4.12. The test results in the simulated field-1

5. DISCUSSION

In this work we studied the global localization without a prior position information problem. This is known to be a very hard task for autonomous mobile robots. We have chosen Robot Soccer as a test bed for our proposed solutions. This is a challenging test bed for the applications designed and optimized for autonomous mobile robots and requires fast, and accurate methods. Here, the perceived sensor data are often inadequate, imprecise and even distorted as a consequence of low computational complexity and memory requirements.

In this work, we studied several well known methods in this domain, their benefits, and failures. The simplest localization method depending on the range and bearing data is triangulation, which uses geometry to compute a single point that is closest to the current location. But in real world applications a robot can never know where it is exactly because of the uncertainty in its sensors, and the environment. Consequently, several different approaches which estimate the position of robot probabilistically were introduced to integrate this uncertainty into the solutions. Built on top of ML and MCL, the R-MCL algorithm is proposed as a fast, reliable, computationally inexpensive and resource efficient solution to the global localization problem, in environments such as the Robocup Games and the Technical Challenges which require very high accuracy and speed, robustness against noise, and insufficient data, and fast recovery from kidnapping.

In Table 5.1, the well-known single localization methods analyzed in this work are categorized according to several useful criteria. Comparisons of several methods including Kalman Filter (KF), Markov Localization (ML), Monte Carlo Localization (MCL), Sensor Resetting Localization (SRL), Adaptive MCL (A-MCL), Mixed-MCL (M-MCL), and Markov Localization-Extended Kalman Filter (ML-EKF) are based on the work in (Gutmann and Fox, 2002). The rest of the methods that are analysed in this table are Multiple Hypothesis Localization (MHL) (Kristensen and Jensfelt, 2003), Kullback-Leibler Distance-Sampling (KLD-S) (Fox, 2003), Simple Localization (SLOC)

Table 5.1. Comparison of Single Localization Methods

Method	Capability of global localization	Accuracy	Speed	Memory usage	Robustness to noise	Fast recovery from kidnapping
KF	no	H	H	L	L	L
ML	yes	L**	M	H**	H	H
MCL	yes	M**	M	H**	M	M
SRL1***	yes	M**	M	M**	M	L
SRL2***	yes	M**	M	M**	L	H
A-MCL	yes	M**	M	M**	XH	H
M-MCL	yes	M**	M	M**	H	H
ML-EKF	yes	M**	M	H**	H	H
MHL	yes	M**	M	H**	H	H
KLD-S	yes	M**	M	H**	H	H
SLOC	yes	M**	M	H**	L	M
Fuzzy*	yes	L**	M	H**	H	H
GEO	yes	H**	H	L	L	L
R-MCL	yes	M**	H	H**	H	H

*Fuzzy method is the method implemented in (Köse *et al*, 2003).

** These are grid based and sample based methods. So accuracy and memory usage changes with the cell size, and the number of samples used. But they still remain in acceptable ranges.

***SRL1 and SRL2 differ in their wish to accept additional samples on each noisy observation. This leads fast recovery from kidnapping but increase noise and decrease accuracy.

(Çelik, 2005), Fuzzy Localization (Fuzzy), Geometrical Localization (GEO), and R-MCL. Detailed information about these methods could be found in Section 2.2. The comparisons have relative values as H for high, M for Medium and L for Low. First comparison item is the capability of global localization. Among these methods only KF can not localize globally. Accuracy is the accuracy of the resultant position. If the method is a grid based method, the accuracy is evaluated as *Low* and if it is a sample based approach, its accuracy is evaluated as $High$, since higher accuracy could be gained using samples rather than large sized grids. Unfortunately using large number of grids and samples increases memory usage. These also affect speed, since as the number of samples or grids which should be integrated in the computations increase, also computational time increase. So KF which do not use any samples or grid cells have high speed and low memory usage, and is very accurate in local localization. Unfortunately it fails in global localization, and kidnapping, as stated before. Generally, sample and grid based methods are robust to noise, when they use enough number of samples or grid cells. Adaptive methods could have lower costs, and high accuracy as their sample sizes differ in different cases. But the parameter sets should be carefully chosen to obtain the best results. According to the criteria in Table 5.1, R-MCL is one of the best methods in the literature. Its success could be increased by using adaptive number of samples and grid cells.

In Table 5.2, the multi-robot localization methods which are studied in this work are compared according to several criteria. These methods are Collaborative Probabilistic Constraint-based Landmark Localization (CPCBL) (Stroupe and Balch, 2002), Representing Hierarchical POMDPs as DBNs for Multi-scale Robot Localization (POMDP) (Theocharous *et al*, 2004), Robust Multi-robot Object Localization Using Fuzzy Logic (Fuzzy) (Canovas *et al*, 2004), Cooperative MCL (CMCL) (Fox *et al*, 2000, Fox *et al*, 1999a), Distributed Multi-Robot Localization (DMRL) (Roumeliotis and Bekey, 2000, Roumeliotis and Bekey, 2000, Roumeliotis and Rekleitis, 2004), Collaborative Multi-Robot Active Localization (CMRAL) (Jones and Shel, 2004), Cooperative Positioning System (CP) (Kurazume and Hirose, 2000), Ego-Centric Approach (EGO) (Howard *et al*, 2003), and CR-MCL, which are described in detail in Section 2.3. The comparisons have relative values as H for high, M for Medium and L for Low. e.g. if the

Table 5.2. Comparison of Multi Robot Localization Methods

Method	Capability of global localization	Accuracy	Speed	Memory usage	Robustness to noise	Comm. cost
CMCL	yes	H	M	M	M	M
POMDP	yes	L	M	H	L	L
Fuzzy	yes	L	M	H	M	H
CPCBL	yes	H	M	L	L	L
DMRL	yes	H	H	L	L	L
CMRAL	yes	H	M	M	M	H
CP	yes	L	L	M	M	L
EGO	no	H	L	H	H	H
CR-MCL	yes	H	H	M	H	L

algorithm uses samples its memory usage is rated as high, if it uses KF and does not transfer samples, its communication cost is low. In these comparisons, using samples is assumed to increase the accuracy, but also increase the cost, and memory usage. Using grid cells is also costly but not as much as using samples, provided that the number of grid cells is smaller than the number of samples. Hence this approach is also faster than sample based algorithms. Additionally, grid based methods have less accuracy than sample based algorithms. Notice that, these assumptions, and evaluations are based on the general parameter sets studied in this work. A sample based algorithm like SLOC which uses very small number of samples (one sample for each observation) would be very fast, and less costly in terms of computational, and memory usage when compared to a grid based algorithm.

In Table 5.2, the first item is the capability of global localization. Among these methods only the Egocentric approach is designed only for relative positioning. The rest of the methods are capable of global localization with different accuracies. The accuracy of the grid based methods is evaluated as *Low*, and the accuracy of the

sample based methods as $High$. Speed is related to the number of grids and samples used which bring a high burden in terms of the computational cost. So the speed of the methods using high number of samples like Egocentric method are rated as Low. The CP method which requires the action of some team members while others are standing still as observers, is also rated as a slow method. The cost of memory usage is directly proportional to the number of samples and grids that are used by the methods. Some methods like POMDP, which do not use any samples or grids are also known to have $High$ memory usage. The Kalman Filter based methods generally have low robustness to noise. The Ego-Centric method which uses a very large number of samples and CR-MCL which is tested under high noise levels are evaluated with $High$ robustness to noise. The rest of the methods are evaluated as $Medium$. Most of the methods use large number of grid cells and samples to represent the shared information, and some of the methods even transfer these via communication which increase communication cost. CR-MCL seems to have better values in all of the criteria, and is a good choice for especially real time applications, as it is fast, cheap in communication, memory and computational cost, and accurate and robust, as well. CR-MCL is also scalable, whereas most of these methods are not, so it can be used in real-time applications, which brings a good contribution to the domain. The multi-robot localization methods studied in this work are not compared with each other in the referenced works. They are just tested against odometry. Furthermore to the best of our knowledge, there is no well-known data set as in the single-robot case, to compare the methods. Therefore, unlike the other relevant works, CR-MCL is not tested against results based on only odometry, but the non-collaborative version of itself, which is a much more challenging opponent. This kind of testing is a better indicator of performance.

The are several drawbacks in the proposed algorithms, beside their advantages. These are the usage of fixed number of cells in ML, and relying on a fixed threshold for the switch between ML and MCL. As a future work, adaptive number of cells could be used. The switch between ML and MCL could also be based on multiple criteria, even a fuzzy inference system could be used for this purpose. This work is also flexible enough to be extended to solve the multi robot object tracking problem.

6. CONCLUSIONS

In this work, the R-MCL algorithm is proposed as a fast, reliable, computationally inexpensive and resource efficient solution to the global localization problem, in environments like the Robocup Games and the Technical Challenges which require very high accuracy and speed, robustness against noise and insufficient data, and fast recovery from kidnapping. R-MCL is a hybrid approach which aims to combine the ML and MCL methods, to make use of the advantages of both, and overcome the disadvantages. The idea behind this algorithm is to converge to a part of the environment by using a coarse 2-D grid based ML and in this local area, call the MCL to find the current position estimation of the robot in a fast, robust and accurate manner. Starting with no prior information about its position, the robot uses ML until the possibilities for its current location are below a threshold. Then samples are thrown in these locations and MCL is called to run with this sample set. MCL is active as long as the confidence of the robot about its location is above a threshold. Then the ML module becomes active again until the robot is confident enough to call MCL again.

The method has been shown to be very robust and fast and requiring less computational power and memory compared to similar approaches and is accurate enough for high level decision making which is vital for robot soccer. Besides it is flexible, simple to implement and can cover the environment with less amount of samples then similar works. It is especially designed for working with imprecise and inadequate sensor data. It outperforms many of the other methods especially in case of recovery from kidnapping problem. It performs well in both offline tests and tests on the robot, outperforms the methods implemented in our project and keeps in a satisfactory range when compared with the results of other methods in (Gutmann and Fox, 1998, Kristensen and Jensfelt, 2003). It has been outperformed by A-MCL, which is an adaptive method, in the high levels of noise and sparsity. Notice that all of the thresholds of R-MCL are fixed in these tests, although the noise and sparsity levels vary on purpose or depending on the lighting and other environmental conditions. Therefore using adaptive thresholds based on the confidence level of the robot and the current measurements might

improve the success in case of high noise and sparsity or varying lighting conditions.

After solving the single self-localization problem, we have developed a novel collaborative localization method, *Collaborative Reverse Monte Carlo Localization (CR-MCL)* for a team of robots, where the shared data are represented as grid cells, fused and integrated into local belief sets by our hybrid self-localization method, R-MCL. It is tested on both a simulator and on real robots and shown to be fast, robust and accurate, and avoids single point failure, and suitable to real-time robotic activities since it is robust under high noise and sparsity. The real power of the method comes from its hybrid nature. It uses a grid based approach to handle detections which can not be accurate in real-time applications, and sample based approach in self-localization which improves its success, although it uses lower amount of samples compared to similar sample based methods. It could be extended to be used in holding relative position information of the team members, and the track of the mission related objects like ball, as well.

There is an ongoing project to improve and use this method in the real-time robot soccer games.

REFERENCES

Akın, H.L., A. Topalov, and O. Kaynak, 2001, "Cerberus 2001 Team Description", *Robocup 2001: Robot Soccer World Cup V* , Birk, A., Coradeschi, S., and Tadokoro, S. (Eds.), LNAI 2377, pp.689-692, Springer Verlag.

Betke, M., and L. Gurvits, 1994, "Mobile robot localization using landmarks", *In Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 135–142.

Betke, M., and L. Gurvits, 1997, "Mobile robot localization using landmarks", *IEEE Transactions on Robotics and Automation*, 13(2):251–263.

Buschka, P., A. Saffiotti, and Z. Wasik, 2000, "Fuzzy Landmark-Based Localization for a Legged Robot" *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)* , pp. 1205–1210.

Burgard, W., D. Fox, D. Hennig, and T. Schmidt, 1996, "Estimating the Absolute Position of a Mobile Robot Using Position Probability Grids," *Institut für Informatik III. , Universität Bonn, Proc. of the Fourteenth National Conference on Artificial Intelligence (AAAI-96).*

Canovas, J-P., LeBlanc, K. , and A. Saffiotti, 2004, "Robust Multi-Robot Object Localization Using Fuzzy Logic", *In: D. Nardi, M. Riedmiller and C. Sammut (eds) RoboCup 2004: Robot Soccer World Cup VIII. Springer-Verlag*, Germany.

Çelik, B., 2005, "S-LOC and MY ENVIRONMENT: A New Localization System for Autonomous Robots", *M.Sc. Thesis*,Bogazici University.

Cerberus Soccer Team, 2005, http://robot.cmpe.boun.edu.tr/aibo/home.php3.

Crisman, Z., E. Curre, C. T. Kwok, L. Meyers, N. Ratliff, L. Tsybert, and D. Fox,

2002, "Team Description: UW Huskies-01", *RoboCup 2001: Robot Soccer World Cup V*, Springer-Verlag, Seattle, Washington, Lecture Notes in Computer Science Series, Vol. 2377, pp 721–724.

Fox, D., W. Burgard, H. Kruppa, and S. Thrun, 1999a, "A Monte Carlo Algorithm for Multi-Robot Localization", *CMU-CS-99-120*,

Fox, D., W. Burgard, and S. Thrun, 1999b, "Markov Localization for Mobile Robots in Dynamic Environments", *Journal of Artificial Intelligence Research*, Vol. 11, pp. 391-427.

Fox, D., W. Burgard, F., Dellaert, and S. Thrun, 1999c, "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots", *Proc. National Conference on Artificial Intelligence (AAAI'99).*

Fox, D., W. Burgard, F., Dellaert, and S. Thrun, 2000, "A Probabilistic Approach to Collaborative Multi-Robot Localization", *In Special issue of Autonomous Robots on Heterogeneous Multi-Robot Systems*, 8(3):325–344.

Fox, D., 2003, "Adapting the Sample Size in Particle Filters Through KLD-sampling" *International Journal of Robotics Research,* 22:985–1003.

Gutmann, J.S., W. Burgard, D. Fox, and K. Konolige, 1998, "An Experimental Comparison of Localization Methods", *In proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS98)*, Victoria, Canada.

Gutmann, J.S., and D. Fox, 2002, "An Experimental Comparison of Localization Methods Continued", *In Proc. of the 2002 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'02)*, Lausanne, Switzerland, pp.454–459.

Gutmann, J. S., 2002, "Markov-Kalman Localization for Mobile Robots", *Int. Conf. on Pattern Recognition (ICRP)*, Vol. 2, No. 2, pp.601–604.

Hightower, J., and G. Borriello, 2001, "A Survey and Taxonomy of Location Systems

for Ubiquitous Computing", *Technical Report*, UW-CSE Tech Report #01-08-03.

Howard, A., M.J. Mataric, G.S. Sukhatme, 2003, "Putting the 'I' in 'Team': An Ego-Centric Approach to Cooperative Localization", *In Proceedings of the International Conference on Robotics and Automation (ICRA 2003)*, IEEE, vol. 1, pp. 868-874, Taipei, Taiwan.

Jones, C.V., and D. A. Shel, 2004, "Collaborative multi-robot active localization", *University of Southern California, Los Angeles, Center for Robotics and Embedded Systems* ,http://www-robotics.usc.edu/ dshell/mcl/res/report.pdf

Kalman, R. E., 1960, "A New Approach to Linear Filtering and Prediction Problems", *Transactions of the ASME–Journal of Basic Engineering*, Vol. 82, Series D, pp.35–45.

Kaplan,K., B. Çelik, T. Meriçli, C. Meriçli, and H. L. Akın, 2006, "Practical Extentions to Vision-Based Monte Carlo Localization Methods for Robot Soccer Domain", *RoboCup 2005: Robot Soccer World Cup IX,* A. Bredenfeld, A. Jacoff, I. Noda, Y. Takahashi (Eds.), LNCS Vol. 4020, pp. 420 - 427.

Köse, H., S. Bayhan, and H. L. Akın, 2003, "A Fuzzy Approach to Global Localization in the Robot Soccer Domain" *IJCI Proceedings of International XII Turkish Symposium on Artificial Intelligence and Neural Networks (TAINN 2003),* ISSN 1304-2386, Vol:1, No:1, pp.1–7.

Köse, H., and H. L. Akın, 2004, "Experimental Analysis and Comparison of Reverse-Monte Carlo Self-Localization Method", *Proceedings of CLAWAR/EURON Workshop on Robots in Entertainment, Leisure and Hobby,* Vienna, Austria, pp.85–90.

Köse, H., and H. L. Akın, 2005, "Robots from nowhere", *RoboCup 2004: Robot Soccer World Cup VIII*, Daniele Nardi, Martin Riedmiller, Claude Sammut, *etal* (Eds.), LNCS, Vol. 3276, pp.594–601.

Köse, H., and H. L. Akın, 2006, "A Fuzzy Touch To R-MCL Localization Algorithm", *RoboCup 2005: Robot Soccer World Cup IX,* A. Bredenfeld, A. Jacoff, I. Noda, Y. Takahashi (Eds.), A. Bredenfeld, A. Jacoff, I. Noda, Y. Takahashi (Eds.), LNCS Vol. 4020, pp.420–427.

Köse, H., B. Çelik, and H. L. Akın, 2006, "Comparison of Localization Methods for a Robot Soccer Team", *International Journal of Advanced Robotic Systems,* Vol. 3, No. 4, pp.295–302.

Köse, H., and H. L. Akın, 2007, "The Reverse Monte Carlo Localization Algorithm", *Robotics and Autonomous Systems,* (Accepted).

Kristensen S., and P. Jensfelt, 2003, "An Experimental Comparison of Localisation Methods, the MHL Sessions", *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'03),* pp.992–997.

Kurazume R., and S. Hirose, 2000, "Collaborative Probabilistic Constraint Based Landmark Localization", *Proceedings of the 2002 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems EPFL*, Lausanne, Switzerland, pp. 447–452.

Maybeck, P. S., 1990, "The Kalman Filter: An Introduction to Concepts", I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*, Springer-Verlag, pp.194–204.

Lenser, S., and M. Veloso, 2000, "Sensor Resetting Localization for Poorly Modelled Mobile Robots", *Proc. ICRA 2000*, IEEE, Vol. 2, pp.1225–1232.

Leonard, J. J., and H. F. Durrant-Whyte, 1991, "Mobile robot localization by tracking geometric beacons", *IEEE Trans. Robotics and Automation*, 7(3):376-382.

Ribeiro, I. and A. Saffiotti (Eds), 2002, "Lecture Notes from European Summer School on Cooperative Robotics", 2-7 September.

Sony 4Legged Robot League, 2005, "Soccer Rules", http://www.tzi.de/4legged/pub/Website/History/Rules2005.pdf.

Robocup Organization, 2005a, http://www.robocup.org/.

Roumeliotis, S.I., and G.A. Bekey, 2000, "Distributed Multi-Robot Localization", *Distributed Autonomous Robotic Systems 4*, , Springer Verlag, pp.179–188.

Roumeliotis, S.I., and G.A. Bekey, 2002, "Distributed Multi-Robot Localization", *IEEE Transactions on Robotics and Automation*, 18(5), pp.781–795.

Roumeliotis, S.I., and I.M. Rekleitis, 2004, "Propagation of Uncertainty in Cooperative Multirobot Localization: Analysis and Experimental Results", *Autonomous Robots*, 17(1), pp.41–54.

Saffiotti, A., 2000, "The 'Team Sweden' Entry at RoboCup 2000".

Saffiotti, A., A. Bjorklund, S. Johansson, and Z. Wasik, 2002, "Team Sweden", *RoboCup 2001: Robot Soccer World Cup V*, Springer-Verlag, Seattle, Washington, Lecture Notes in Computer Science Series, Vol. 2377, pp.725-729, 2002.

Saffiotti, A.(Organizer), 2002a, *Proceedings Workshop WS7 Cooperative Robotics, IROS 2002, IEEE/RSJ International Conference on Intelligent Robots and Systems.*

Schulz,D., and W. Burgard, 2001, "Probabilistic State Estimation of Dynamic Objects with a Moving Mobile Robot," *Robotics and Autonomous Systems*, 34, pp.107–115.

Sony's AIBO product, http://www.aibo.com/.

Sony Four-Legged Robot League, 2005, http://www.tzi.de/4legged/.

Stroupe, A. W., and T. Balch, 2002, "Collaborative Probabilistic Constraint Based Landmark Localization", *Proceedings of the 2002 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems EPFL*, Lausanne, Switzerland, pp. 447–452.

Stroupe, A.W., K. Sikorski, and T. Balch, 2003, "Constraint-Based Landmark Local-

ization", *RoboCup 2002: Robot Soccer World Cup VI*, Springer-Verlag, Fukuoka, Busan, Lecture Notes in Computer Science Series, Vol. 2752, pp.8–24.

Theocharous, G., K. Murphy, and L. P. Kaelbling, 2004, "Representing hierarchical POMDPs as DBNs for multi-scale robot localization", *In Proceedings of the International Conference on Robotics and Automation (ICRA 2004).*

Thrun, S., D. Fox, W. Burgard, and F. Dellaert, 2001, "Robust Monte Carlo Localization for Mobile Robots", *Artificial Intelligence*, Elsevier, Vol. 128, pp.99–141.

Thrun, S., W. Burgard, and D. Fox, 2005, "Probabilistic Robotics", MIT Press, Cambridge, MA,ISBN 0-262-20162-3.

Welch, G., and G. Bishop, 2006, "An Introduction to the Kalman Filter", UNC-Chapel Hill, TR 95-041.

REFERENCES NOT CITED

Akın, H. L., Pavlova, P., and Yildiz, O. T., 2002, "Cerberus 2002", *Robocup 2002: Robot Soccer World Cup VI* , The 2002 International Robocup Symposium Pre-Proceedings, June 24-25, Fukuoka, pp.448.

Borenstein, J., H.R. Everett, and L. Feng, 2001, "Navigating Mobile Robots, Systems and Techniques", A. K. Peters, Ltd., Wellesley, MA.

Corne, D.W, Deb, K., Fleming, P.J., and Knowles, J.D., 2003, "The Good of the Many Outweights the Good of the One: Evolutionary Multi-Objective Optimization", *coNNectionS* ,1(1): 9-13, ISSN 1543-4281.

Dias, M. B., and A. Stenz, 1999, "A Free Market Architecture for Coordinating Multiple Robots", CMU-RI-TR-99-42, December.

Dias, M. B., and A. Stenz, 2001, "A Market Approach to Multirobot Coordination", CMU-RI-TR-01-26, August.

Dietel, M., J.S. Gutmann , and B. Nebel, 2001, "Cooperative Sensing in Dynamic Environments", *In Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001)* , Maui, Hawaii.

Dissanayake, G., P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba, 2001, "A Solution to the Simultaneous Localization and Map Building (SLAM) Problem", *IEEE Transactions on Robotics and Automation* ,Vol. 17, No. 3, June.

Gerkey, P.B., and M.J. Mataric, 2002, "Sold!: Auction methods for multi-robot coordination", *IEEE Transactions on Robotics and Automation*, special issue on Advances in Multi-Robot Systems, 18(5), pp.758–786.

Guivant, J. E., and E. M. Nebot, 2001, "Optimization of the Simultaneous Localization and

Map-Building Algorithm for Real-Time Implementation", *IEEE Transactions on Robotics and Automation* ,Vol.17, No.3, June, pp.242–257.

Kaplan, K., 2001, "Design and Implementation of fast controllers for Mobile Robots", *Master Thesis* ,Vol. 17, No. 3, June.

Köse, H., C. Meriçli , K. Kaplan, and H. L. Akın, 2003, "Genetic Algorithms based Market-Driven Multi-Agent Collaboration in Robot Soccer Domain", *Proceedings of 2003 FIRA Robot World Congress*, Vienna.

Köse, H., C. Meriçli , K. Kaplan, and H. L. Akın, 2003, "All Bids for One and One Does for All: Market-Driven Multi-Agent Collaboration in Robot Soccer Domain", *ISCIS XVIII The Sixteenth International Symposium on Computer and Information Sciences (ISCIS 2003)*, Springer-Verlag, Lecture Notes in Computer Science Series,2003.

Köse, H., U. Tatlıdede, C. Meriçli , K. Kaplan, and H. L. Akın, 2004, "Q-Learning based Market-Driven Multi-Agent Collaboration in Robot Soccer", *XIII Turkish Symposium on Artificial Intelligence and Neural Networks TAINN 2004*, June 10-11, Izmir, Turkey, pp.219-228.

Köse, H., K. Kaplan, C. Meriçli , U. Tatlıdede, and H. L. Akın, 2005, "Market-Driven Multi-Agent Collaboration in Robot Soccer Domain", in V. Kordic, A. Lazinica and M. Merdan (Eds.), *Cutting Edge Robotics*, pp.407–416, pIV pro literatur Verlag, 2005.

Nagatani, K., H. Choset, and T. Thrun, 1998, "Towards Exact Localization without Explicit Localization with the Generalized Voronoi Graph", *In IEEE International Conference on Robotics and Automation*, Luven, Belgium, May.

Schmitt, T., R. Hanek, S. Buck, and M. Beetz, 2001, "Cooperative Probabilistic State Estimation for Vision-based Autonomous Mobile Robots", *In Proc. of the IEEE Intl. Conf. on Intelligent Robots and Systems (IROS 2001)*, Maui, Hawaii.

Schulz, D., and W. Burgard, 2001, "Probabilistic State Estimation of Dynamic Objects with a Moving Mobile Robot", *Robotics and Autonomous Systems*, Elsevier, 34, pp. 107-115.

Schulz, D., W. Burgard, D. Fox, and A.B. Cremers, 2001, "Tracking Multiple Moving Objects with a Mobile Robot", *In Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, Kauwai, Hawaii.

Tatlıdede, U., K. Kaplan, H. Köse, and H. L. Akın, 2005, "Reinforcement Learning for Multi-Agent Coordination in Robot Soccer Domain", *AAMAS'05, Fifth European Workshop on Adaptive Agents and Multi-Agent Systems*, Paris, March 21-22 2005. (Accepted)

Thrun S., 2002, "Robotic Mapping: A survey", CMU-CS-02-111.

Veloso, M., S. Lenser, D. Vail, M. Roth, A. Stroupe, and S. Chernova, 2002, "CMPack-02: CMU's Legged Robot Soccer Team", Carnegie Mellon University, Pittsburgh, October.

Zlot, R., A. Stenz, M. B. Dias, , and S. Thayer, 2002, "Market-driven Multi-Robot Exploration", Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR02 -02, January.