2010

Robot localization in symmetric environment

Ali Akhavan Malayeri
University of Windsor

Robot Localization in Symmetric Environment

By

Ali Akhavan Malayeri

A Thesis

Submitted to the Faculty of Graduate Studies

through Computer Science

in Partial Fulfillment of the Requirements for

the Degree of Master of Science at the

University of Windsor

Windsor, Ontario, Canada

2010

Your file Votre référence
ISBN: 978-0-494-62738-9
Our file Notre référence
ISBN: 978-0-494-62738-9

Author's Declaration of Originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

The robot localization problem is a key problem in making truly autonomous robots. If a robot does not know where it is, it can be difficult to determine what to do next. Monte Carlo Localization as a well known localization algorithm represents a robot's belief by a set of weighted samples. This set of samples approximates the posterior probability of where the robot is located. Our method presents an extension to the MCL algorithm when localizing in highly symmetrical environments; a situation where MCL is often unable to correctly track equally probable poses for the robot. The sample sets in MCL often become impoverished when samples are generated in several locations. Our approach incorporates the idea of clustering the samples and organizes them considering to their orientation. Experimental results show our method is able to successfully determine the position of the robot in symmetric environment, while ordinary MCL often fails.

Dedication

I dedicate this thesis to my wife Arezoo.

Without whom this, along with many other things, would never have been possible. Thank you, for all your love, faith, patience, support and assistance.

Acknowledgements

I would like to thank my family for never doubting that I would complete this thesis. They always asked by "when" I would complete rather than "if"

I wish to express my thanks to my supervisor, Dr. Dan Wu. This thesis would not have been complete without his expert advice and unfailing patience. I am also most grateful for his faith in this study especially in the sometimes-difficult circumstances in which it was written.

I want to thank my external reader, Dr. Huapeng Wu, my internal reader, Prof. Boubakeur Boufama and my thesis committee chair, Dr. Xiaobu Yuan for spending their time on reviewing this thesis and for all their help, support, interest, and valuable hints.

My friends helped me solve many difficult problems during my research. I want to thank all of them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

In a complex environment such as symmetric environment, localization and navigation of a mobile robot- autonomously toward a goal is a very fascinating problem. A mobile robot can navigate mainly using a global map constructed from sensor information but before that, a robot need to localize itself based on matching local or global sensor information to this map and then decides its behavior subsequently based on the matching results.

The mobile robot localization problem is introduced in many different ways [1, 2]. The simplest localization problem is *position tracking* [1, 3, 4, 5]. Here the initial pose of robot is known, and the problem is to compensate small, incremental errors in a robot's odometry. More challenging is the *global localization problem* [6], where a robot is not told its initial pose, but instead has to determine it from scratch. The global localization problem is more difficult, since the error in the robot's estimate cannot be assumed to be small. Even more difficult is the *kidnapped robot problem* [7], in which a well-localized robot is transported to some other place without being told. This problem differs from the global localization problem in that the robot might firmly believe to be somewhere else at the time of the kidnapping. The kidnapped robot problem is often used to test a robot's ability to recover from catastrophic localization failures.

Many algorithms have been proposed for these problems. For example, Kalman filter [8, 9, 10, 11], Grid localization [12, 13, 14], Monte Carlo localization [15, 16, 17] and some hybrid approaches [18, 19]. Undoubtedly, one of the most popular algorithms is the Monte Carlo localization (MCL). MCL solves the global localization and kidnapped robot problem in a highly robust and efficient way [20]. In contrast to Kalman filter based techniques which only work well for unimodal distributions, MCL is able to represent multi-modal distributions and can globally localize a robot. MCL is

more accurate than Grid localization with a fixed cell size. Moreover, it is surprisingly easy to implement, which makes them an attractive paradigm for mobile robot localization.

The key idea of MCL is to represent the belief by a set of samples (*particles*), drawn according to the posterior distribution over robot poses. In other words, MCL simply represents the posteriors by a random collection of weighted particles which approximates the desired distribution [20]. However, all these proposed methods have particularly become unreliable in case of dynamic or symmetric environments and the localization problem is then become more challenging in these environments.

1.1 Motivation

Among many localization techniques, MCL has become arguably the most popular approach to date. By using a sampling-based representation, MCL has several key advantages over earlier work in this field. For example, it reduces the amount of memory required compared to grid-based Markov localization and in contrast to existing Kalman filtering, it is able to represent multimodal distributions. It is also surprisingly easy to implement, which makes that an attractive paradigm for mobile robot localization.

However, there are some disadvantages too. For example, the standard MCL technique may fail during localization when there are *similar* locations in the robot's environment such as Robocop soccer field. The problem arises when samples are generated according to the posterior distribution (as is the case in MCL), they may represent the multimodal distributions that often arise during the localization in symmetric environment. This might be undesirable in symmetric environments, where multiple hypotheses have to be tracked for extended periods of time. However, although the MCL method is able to initially represent a multimodal belief distribution, it is unable to maintain it especially when the environment is highly symmetric [50].

In this thesis, we propose a novel approach, called "robot localization with known orientation" which is based on Monte Carlo Localization framework. This proposes method represents an extension to the MCL algorithm, when environment is highly symmetric. Normally when executing MCL in symmetric environment, after iterating several steps, particles are accumulated in several locations. We use a simple clustering algorithm to separate the points (*particles*) into different clusters. We then compare each particle and consider if it is either allocated to an existing cluster or assigned to a newly created cluster, depending on its orientation. The comparison is performed based on our robot's orientation which is initially postulated parallel with the direction of the x-axis.

Once clusters are generated, we utilized them to localize our robot in the environment. However, according to the symmetric feature of our environment, three possible situations are considered as below:

1) Case one happens when most particles are accumulated around the true location of the robot. This accumulation represents that the robot is successfully localized in its actual location. However, the key point in this scenario is although particles become clustered all around the true position of robot, there is no such a condition to confirm that localization process is ended and now is time for robot to start position tracking.

2) Case two happens when particles are accumulated contrariwise. This is a situation when particles often too quickly converge to a single, high likelihood pose and then ignoring the possibility that the robot might be located in somewhere else. The resultant cluster in this case represents the location of robot in opposite side. For example in Robocop soccer field, when the robot is located in upper corner, particles will show this location in bottom side. This reverse situation is created because particles have the opposite orientation. Thus, when the value of the robot's orientation is equal to 45° ($\theta = 45°$), we have a set of particles that their orientation value are equal to 225°. This

situation therefore considers as a localization problem in symmetric environment and we will evaluate it in our propose method.

3) In third case, the distribution of robot pose is usually multimodal due to the symmetry of the environment and ambiguous detected features. In this case, some particles are accumulated around the true location of the robot when some other particles are accumulated contrariwise and represent the location of robot in opposite side. Generally speaking, this case is the mixture of the two previous cases, on one hand some particles show the true location of robot and on the other hand some other particles indicate this position vice versa. This case is also considered as a localization problem in symmetric environment because the position of robot is shown in two different locations. Same as previous case, this problem will also be resolved with our propose solution for mobile localization in symmetric environment.

We applied our method in all these three situations and clustered the particles based on the orientation. In order to do that, we take the MCL particle set as input that needs to be clustered. We then initialized the orientation of robot as a representative point and then we create our clusters according to this value. The value of representative point is initially postulated parallel with the direction of the *x-axis* ($\theta = \pi$) and the threshold of dissimilarity $\Theta = 20^{\circ}$ which is derived from the experiment has been considered as input. Calculating dissimilarity measures between current particle and every cluster to find a minimum one is considered as a next step. If the minimum measure was larger or smaller than Θ, a new cluster that contains current particles will be created. Otherwise, the considered particle will be assigned to the existing cluster which has a minimum dissimilarity measure to it.

According to this implementation, we will be able to distinguish those clusters that have the same orientation with robot and those with other orientation, including the opposite one. This solution can be applied to all the cases that we mention above as different possible situations in symmetric environment. Take for example, when we apply

our method in case three, we can easily distinguish that which cluster consist the true orientation and therefore indicating the actual location of robot and which one is showing the virtual location of our robot due to the feature of the symmetric environment.

1.2 Contributions

This thesis is only concerned with the problem of Monte Carlo localization in symmetric environments, particularly in small-scale room with robot equipped with low-cost sensors. In this thesis, we introduce a cluster-based extension to MCL algorithm called "robot localization with known orientation". In this method, although clustering plays a very important role but our main concern is based on the orientation. Ordinary MCL can fail if the map is symmetrical, however, we propose a method based on this significant problem and in next chapters, we demonstrate that this method is valid and reliable. Experiments have been conducted with both simulated data as well as data obtained from a real robot. The results show that our algorithm is able to successfully determine the position of the robot in these environment, while ordinary MCL often fails

1.3 Outline

This thesis is organized as follows:

Chapter 2: Localization and Probabilistic Framework. This chapter introduces the problem of robot localization and describes different instances of the problem. We discuss a framework that we can use to formalize the uncertainty and beliefs in the localization problem. Furthermore, we approach the localization problem from a probabilistic point of view and look at different solutions to the localization problem that implement this framework.

Chapter 3: The Proposed Method. The proposed method is presented in detail in this chapter. The statement of the problem and the general description of our method have been discussed in this chapter. It is also followed by which clustering algorithm is chose and how to combine it with MCL.

Chapter 4: Implementation and experiment results. Details information of the implementation and the experimental results has been discussed in this chapter. Then, experimental results are presented that this new extension to the MCL algorithm successfully localizes in symmetric environments where ordinary MCL often fails.

Chapter 5: Conclusion and future work. We summarize the presented work with concluding and we also present ideas and possibilities for future research.

Chapter 2

Mobile Robot Localization in Probabilistic Framework

In this chapter we will take a look at robot localization and argue it in probabilistic framework. In Section 2.1We will discuss the general problem and review the different type of information to which a robot has access for localization. In Section 2.2 we will discuss features in the environment that a robot can detect. In Section 2.3 Localization in probabilistic framework is discussed. In Section 2.4 we introduce the notion of belief, and formalize the acting and sensing of a robot in probabilistic models and then use these models in section 2.5 to derive a general probabilistic formula for localization. We look to several implementations method in Section 2.6 and discuss how they deal with localization problem.

2.1 Localization problem

The problem of robot localization consists of answering the question "*Where am I?*" from a robot's point of view. This means the robot has to find out its *location* relative to the environment. When we talk about *location*, *pose*, or *position* we mean the x and y coordinates and θ which is the heading direction of a robot in a global coordinate system.

In determining this location, a robot has access to two kinds of information. First one is a priori information, gathered by the robot itself or supplied by an external source as an initialization phase. In general, this information supplied to the robot describes the environment where the robot is driving around. Second is navigational information that the robot gathers from its sensors while navigating through the environment.

In order to navigating through the environment, a robotic vehicle has a driving system which is playing an important role in physical position of robot [25, 33]. The way that driving system changes the location of robot, contains valuable information in

estimating the location. However, in most robotic applications, this value is acquired by odometry sensors. Odometry works by integrating incremental information over the time and by using wheel encoders to count the number of revolutions of each wheel, the robot measures the distance it traveled and its heading direction. Odometry is widely used, because it gives good short-term accuracy, is inexpensive, and allows for very high sampling rates [27, 28]. Thou, due to drift and slippage the integration of the wheel revolutions lead to errors in both traveled distance and orientation [28,42]. These errors accumulate over time and in particular errors in the orientation, cause large positioning errors. Another disadvantage of odometry is its sensitivity to terrain. If the surface the robot is driving on is not smooth, it can result in considerable position errors, since the odometry system cannot detect the irregularities in the terrain. Also differences in wheel diameter can cause position errors that are not detected by odometry measurements [30,40]. Although, odometry causes increasing error in the location estimate, it is the easiest way to access form of position information and therefore it is an important source of information for localization.

Sensing the environment through the sensors is another factor to navigating the environment. These sensors give momentary situation information, called observations or measurements. This information in one hand describe things about the environment of the robot at a certain moment and on the other hand provide the location of the robot that is independent of any previous location[26, 35].This independency has the advantage that the error in the position does not grow unbounded.

2.2 Features of Environment

The feature of our environment is contributed to obtain the measurements data and can be introduced into different model. One of the common types of environments is symmetric environment which is very ambiguous for robot to localize itself successfully through that. Take for example the Robocop soccer field has a very complex symmetrical form for robot to localize itself in actual location. Figure 1.1 illustrates this situation in a robot soccer field.

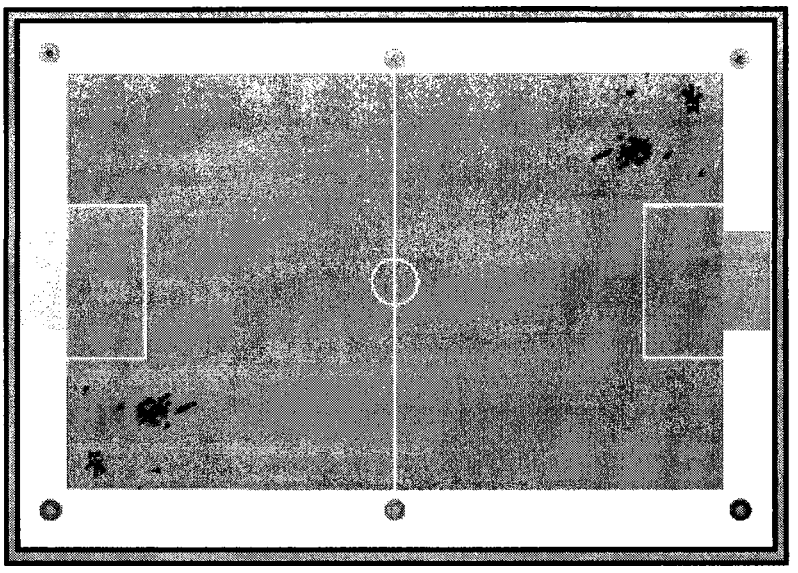


Figure 2.1: Robocop soccer filed

In this case, for any considered locations as a possible location of our robot, we have another match location in other side that represents the location of robot symmetrically.

2.2.1 Landmarks

This complexity can be reduced by adding obstacles or landmarks to our environments. Landmarks are features in the environment that a robot can detect them. Sensor readings from a robot are analyzed for the existence of landmarks and once landmarks are detected, they are matched with priori known information of the environment to determine the position of the robot. Landmarks can be divided into active and passive landmarks. Active landmarks are landmarks that actively send out location information and robot senses the signals sent out by the landmark to determine its position. For example the *GPS* sensor [30], uses information from uniquely coded radio signals sent from satellites.

However, active landmarks in practice often cannot send out their signals in all directions, and thus cannot be seen from all places. Furthermore, active landmarks may

be expensive to construct and maintain. If the landmarks do not actively transmit signals, the landmarks are called passive landmarks. The robot has to actively look for these landmarks to acquire position measurements. Techniques using passive landmarks in determining the position of the robot rely on detection of those landmarks from sensor readings. The detection of landmarks depends on the type of sensor used. For example, in detecting landmarks in images from a vision system, image processing techniques are used.

2.2.2 Map

Another group of localization techniques are map based positioning. These approaches use geometric features of the environment to compute the location of the robot. Examples of geometric features are the lines that describe walls in hallways or offices. Sensor output is then matched with these features. Model matching can be used to update a global map in a dynamic environment, or to create a global map from different local maps [24]. Using this technique to determine the absolute position of a robot has the disadvantage that there needs to be enough sensor information to be matched with the map to come up with a position. Furthermore, techniques for matching sensor data with maps often require large amounts of processing power and sensing [25].

2.3 Localization and Probabilistic

If we look at the localization problem probabilistically, we can say that the robot has a belief about where it is. At any time, it does not consider one possible location, but the whole space of locations. The localization problem consists of estimating the probability density over the space of all locations.

The robot has a belief which is the probability density over all locations $x \in \Xi$, where Ξ is the set of all locations. We denote the belief by *Bel*. Localization can be seen as maintaining the belief,

$$Bel(x_k) = P\,(x_k|d_{0\ldots k}) \qquad (2.1)$$

That is, the probability that the robot is at location x_k at time k, given all information or data $d_{0...k}$ up to that time. The location that gives this probability distribution has the highest probability in the location at which the robot is most likely to be. The goal of localization is to make this belief get as close as possible to the real distribution of the robot location. The real distribution of the robot location has a single peak at the true location and is zero everywhere else. If the robot achieves this goal, then it knows exactly where it is located.

However, in some cases, during the localization, the robot has access to absolute and relative measurements. Relative measurements are measurements that are made by looking at the robot itself only. The robot incorporates these measurements into its belief to form a new belief about where it is. To be able to update the beliefs with the latest measurement information, we need to express measurement information in probabilistic terms. We need to define a probabilistic model for the acting, that is, the relative measurements, and a probabilistic model for the sensing, that is, the absolute measurements.

2.3.1 Acting

A robot performs actions and changes its position in the environment. We define action a_k from a set of possible actions and express the location of the robot changes probabilistically by a transition density as [31, 2]

$$P(x_k | x_{k-1}, a_{k-1}) \tag{2.2}$$

This probability density gives the probability that if at time step $k-1$ the robot was at location x_{k-1} and performed action a_{k-1}, then it ended up at location x_k at time step k. In other words, the transition density describes how the actions of the robot change its location. This density is therefore called the *action* or *motion model*.

Actions contain relative information about the new location of a robot. By given the last location, the robot can estimate its current location based on the performed action. Without the last location, the robot only knows it made a certain move; it is not able to

label an absolute location to the resulting position. In practice we can roughly approximate this transition density from the kinematics and dynamics of the robot. Another option is to have the robot learn the model itself [31, 49].

2.3.2 Sensing

We can also describe the sensing of the robot in probabilistic terms. Let S be the space of all possible measurements coming from a sensor, and let s_k denote an element in S observed at time k. We can describe the probability that a sensor observes s_k from a certain location x_k at time k by the density [31, 2].

$$P(s_k|x_k) \tag{2.3}$$

This is called the *sensor* or *perceptual model*. As with the motion model, the perceptual model is often time-invariant. In that case we can omit the time subscript k.

Unlike the transition density of the acting of the robot, this probability density is difficult to compute. The reason for this is the sometimes high dimensionality of the measurements. Consider for example how complex the probability density is if the measurements come from a camera. The probability density will have to give a probability for each possible camera picture at each possible location, which would require a large amount of computing power.

2.4 Localization Formula

The robot performs an action and this action changes the location of the robot according to the transition density from. Besides this, the robot can also get information from sensing the environment and perhaps extracts features from this sensor information to form a feature vector which is distribute according to the probability distribution from.

The robot now has to update its belief with the new information in order to get the best location estimate.

2.4.1 Belief

Before the robot start acting in the environment it has an *initial belief* of where it is. We model this belief by the prior belief at time step 0, $Bel^{-}(x_0)$. If the robot knows where it initially is, then $Bel^{-}(x_0)$ is a distribution with a peak at the location where the robot knows it is. The goal of the localization becomes to compensate for slippage, drift and possible other noise sources to keep track of the location. This problem called the *position tracking* problem. In the case that the robot does not know where it starts, the initial belief $Bel^{-}(x_0)$ is a uniform distribution. The problem of localization is to make the robot localize itself, not having any idea of where it is. This is described as the *wake-up robot* or *global localization* problem. Finally, in the case that the robot thinks it is at a certain location, but it actually is not there, the initial belief is initialized with a peak at the location where the robot thinks it is. Since it is not actually located there, the robot has to detect this and adjust its belief. This is called this the *kidnapped robot* problem.

Starting with the initial belief the robot starts querying its sensors and performing actions in the environment. The resulting measurements and actions have to be incorporated into the belief of the robot to give it the most up-to-date location estimate. The belief the robot has after it has incorporated the action a_{k-1} executed at step $k-1$, and before it gets a new measurement z_k, is the prior belief,

$$Bel^{-}(x_k) = P(x_k | z_1, a_1, z_2, a_2, \ldots, z_{k-1}, a_{k-1}) \qquad (2.4)$$

Once it has received an absolute measurement z_k at step k, it incorporates this measurement to obtain the posterior belief,

$$Bel^{+}(x_k) = P(x_k | z_1, a_1, z_2, a_2, \ldots, z_{k-1}, a_{k-1}, z_k) \qquad (2.5)$$

2.4.2 Incorporating Acting

Assume the robot has performed an action and wants to include the relative position measurement result of this action into its belief. In equation (2.4) we defined the belief which is the latest action information incorporated, the prior belief $Bel^{-}(x_k)$. We

can rewrite this original definition by utilizing the theorem of total probability and use Markov assumption. The theorem of total probability states that the probability of an outcome is equal to the sum of the probabilities of each of its dependent, partial, outcomes [33]. Using this theorem, we rewrite the definition of the prior belief (2.4) to

$$Bel^{-}(x_k) = \int_{\Xi} P(x_k|x_{k-1}, z_1, a_1, \ldots, z_{k-1}, a_{k-1}) \times$$

$$P(x_{k-1}|z_1, a_1, \ldots, z_{k-1}, a_{k-1})\, dx_{k-1}. \qquad (2.6)$$

This equation expresses that the prior belief of being in state x_k is the sum of the probabilities of coming from state x_{k-1} to state x_k given all the earlier actions and measurements, $P(x_k|x_{k-1}, z_1, a_1, \ldots, z_{k-1}, a_{k-1})$, multiplied by the probability of actually being in state x_{k-1} given all the earlier measurements and actions, $P(x_{k-1}|z_1, a_1, \ldots, z_{k-1}, a_{k-1})$.

The second term of the integral in (2.6) is the probability of being at location x_{k-1} given all information up to step $k-1$; in particular the action performed at step $k-1$. However, the physical location of the robot at step $k-1$ does not depend on the action that is performed at that step. Therefore, we do not have to take a_{k-1} into account when expressing this probability. Using this and the definition of the posterior belief from (2.5), we rewrite (2.6) into

$$Bel^{-}(x_k) = \int_{\Xi} P(x_k|x_{k-1}, z_1, a_1, \ldots, z_{k-1}, a_{k-1}) \times$$

$$P(x_{k-1}|z_1, a_1, \ldots, z_{k-2}, a_{k-2}, z_{k-1}) dx_{k-1}$$

$$= \int_{\Xi} P(x_k|x_{k-1}, z_1, a_1, \ldots, z_{k-1}, a_{k-1}) \times$$

$$Bel^{+}(x_{k-1}) dx_{k-1}. \qquad (2.7)$$

To simplify the expression of the first term of the integral in (2.7) we make a Markov assumption [2, 37], which states that given knowledge of the current state, the past is independent of the future, and vice versa. With knowledge of the previous

location x_{k-1} , it is of no importance how the robot ended up at that location or what it sensed. With this, we have that

$$P(x_k|x_{k-1}, z_1, \ldots, z_{k-1}, a_{k-1}) = P(x_k|x_{k-1}, a_{k-1}). \tag{2.8}$$

The right hand side of this equation is the conditional probability of being in state x_k given knowledge of the previous state and the performed action. We defined this as the action model in (2.2). By substituting the result into (2.7) we obtain an equation that can be used to efficiently incorporate the robot's actions into its belief,

$$Bel^-(x_k) = \int_{\Xi} P(x_k|x_{k-1}, a_{k-1})\, Bel^+(x_{k-1}) dx_{k-1} \tag{2.9}$$

That is, the prior belief of the robot in being at location x_k .

2.4.3 Incorporating Sensing

Assume that the robot has the prior belief $Bel^-(x_k)$, the belief in the location after it has performed its last action. The robot makes a measurement of the environment and extracts a feature vector z_k from this measurement. We want to incorporate this measurement into the prior belief to form the posterior belief as we defined in equation (2.5). With Bayes' rule and the Markov assumption we can rewrite this posterior belief into a computation-ally efficient form.

Bayes' rule [36, 33] explains how the robot has to change its belief when a new measurement arrives. Using Bayes' rule and the definition of the prior belief from (2.4), we can rewrite (2.5),

$$Bel^+(x_k) = \frac{P(z_k|x_k, z_1, a_1, \ldots, z_{k-1}, a_{k-1}) P(x_k|z_1, a_1, \ldots, z_{k-1}, a_{k-1})}{P(z_k|z_1, a_1, \ldots, z_{k-1}, a_{k-1})}$$

$$= \frac{P(z_k|x_k, z_1, a_1, \ldots, z_{k-1}, a_{k-1})\, Bel^-(x_k)}{P(z_k|z_1, a_1, \ldots, z_{k-1}, a_{k-1})} \tag{2.10}$$

That is times the prior belief of being in state x_k, $Bel^-(x_k)$, divided by the probability of observing measurement x_k conditioned on all information so far, $P(z_k|z_1,\ldots,a_{k-1})$.

To make the computations of equation (2.10) less complex, we again make the Markov assumption. In this case we use it to state that a sensor reading only depends on the current state. The sensor reading is not influenced by previous locations of the robot. It does not matter how the robot got at the current location. The probability of observing a measurement is independent of the actions and observations that were made before the robot arrived in its current state. We use this assumption to rewrite the first term in the nominator of (2.10),

$$P(z_k|x_k, z_1, a_1, \ldots, z_{k-1}, a_{k-1}) = P(z_k|x_k) \qquad (2.11)$$

When we make the Markov assumption, the conditional probability of observing measurement z_k given the current state and past actions and observations reduces to the sensor model from (2.3). If we substitute this into (2.10), we obtain

$$Bel^+(x_k) = \frac{P(z_k|x_k)\, Bel^-(x_k)}{P(z_k|z_1,a_1,\ldots,z_{k-1},a_{k-1})} \qquad (2.12)$$

The denominator of this equation is a normalizing constant ensuring that the probability density integrates to 1. This constant is calculated by integrating the numerator over all possible locations x_k [36, 31],

$$P(z_k|z_1, a_1, \ldots, z_{k-1}, a_{k-1}) = \int_{\Xi} P(z_k|x_k)\, Bel^-(x_k) dx_k \qquad (2.13)$$

Equation (2.12) shows how we can express the posterior belief in terms of the prior belief. It also shows how we update the posterior belief to incorporate a new absolute measurement. It is a computationally efficient equation due to the use of the sensor model and the prior belief.

Finally we can combine the derived results into a single localization equation for the posterior belief in the location of a robot taking into account sensing and action information. Substituting equation (2.9) into equation (2.12), the posterior belief becomes

$$Bel^{+}(x_k) = \frac{P(z_k|x_k)\, Bel^{-}(x_k)}{P(z_k|z_1,\ldots,a_{k-1})}$$

$$= \frac{P(z_k|x_k)\ \int_{\Xi} P(x_k|x_{k-1},a_{k-1}) Bel^{+}(x_{k-1}) dx_{k-1}}{P(z_k|z_1,\ldots,a_{k-1})}$$

$$= \eta_k\ P(z_k|x_k)\ \int_{\Xi} P(x_k|x_{k-1},a_{k-1})\, Bel^{+}(x_{k-1}) dx_{k-1} \qquad (2.14)$$

Where η_k is the probability density normalize $P(z_k|z_1,\ldots,a_{k-1})^{-1}$, calculated as in equation (2.13).

2.5 Methods of Implementation

A way of dealing with continuous location spaces is by discretization or factorization of the space [19, 48]. This way of representing the belief is captured by Hidden Markov Models [3, 44]. These are general models in terms of transition and measurement probabilities. A number of methods has been developed using different representations for the discretization [34, 39]. However we only focus one *particle filters* as a one way of discretization and introduce it as below.

2.5.1 Particle Filters

Particle filter represents the posterior distribution $Bel^{+}(x_k)$ by set of random samples drawn from this distribution. Each particle, which is a sample of the posterior distribution, represents a possible state to be estimated at time t. The input of particle filter is the particle set x_{k-1}, along with the most recent control a_k and the most recent measurement z_k. MCL proceeds in two phases:

Robot motion. When the robot moves, MCL generates N new samples that approximate the robot's position after the motion command. Each sample is generated by *randomly* drawing a sample from the previously computed sample set, with likelihood determined by their p values. This value is a numerical weighting factor, analogous to a discrete probability.

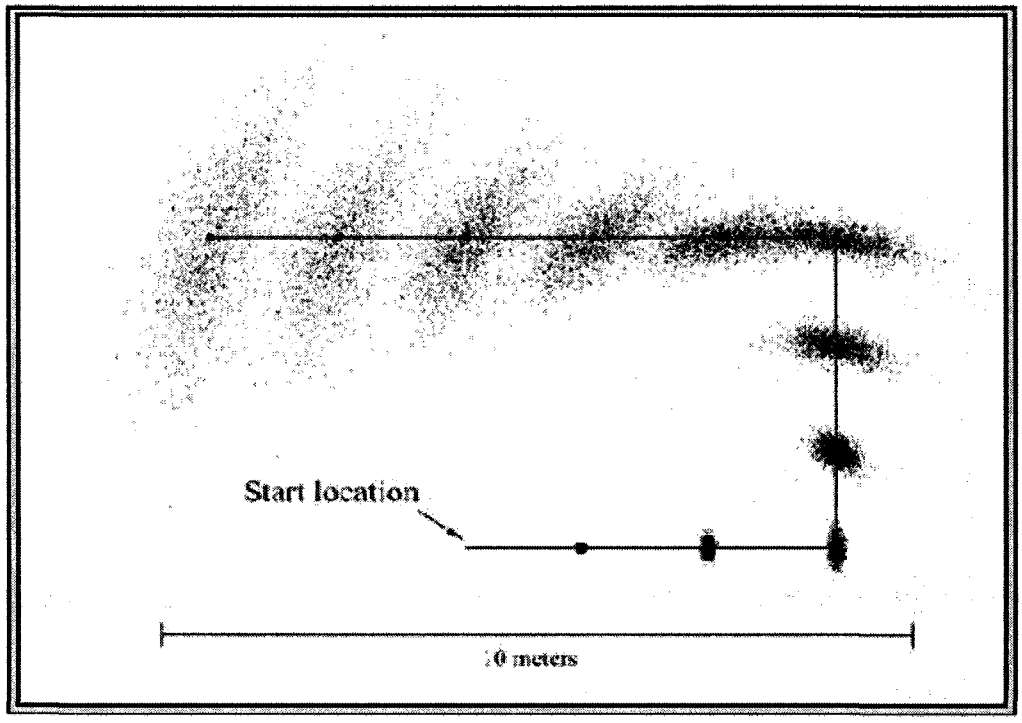


Figure 2.2 Sampling-based approximation of the position belief for anon-sensing robot [45].

Figure 2.2 shows the effect of this sampling technique, starting at an initial known position and executing actions as indicated by the solid line. As can be seen there, the sample sets approximate distributions with increasing uncertainty, representing the gradual loss of position information due to slippage and drift.

Sensor readings describe the formation process by which sensor measurements are generated in the physical world. In fact, it is defined as a conditional probability distribution $P(z_k|x_k, m)$, where x_k is the robot pose, z_k is the measurement at time k, and m is the map of the environment. Figure 2.3 shows a typical range-scan obtained in a corridor with a mobile robot.

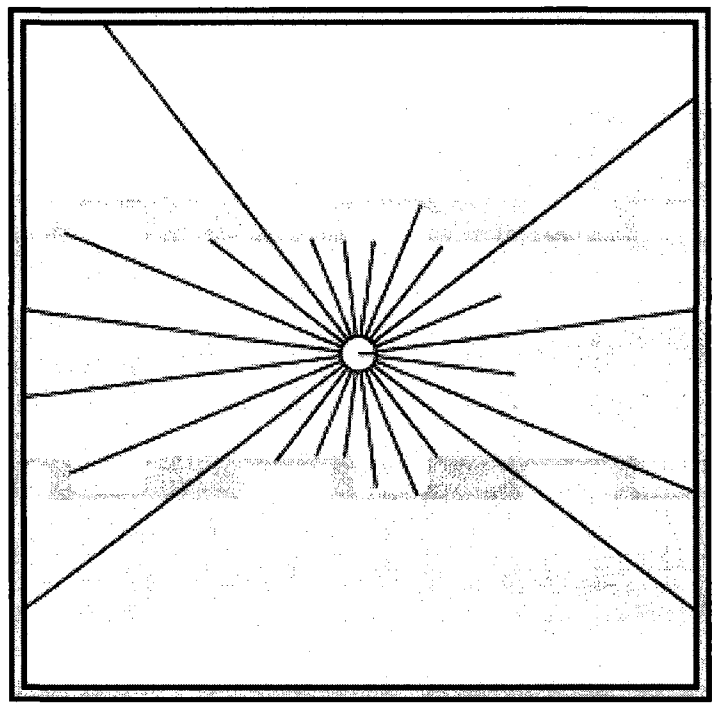


Figure 2.3: Typical scanner of a robot in its environment. [45]

However, the sensors equipped on Create for detecting the external environment are really limited. In our experiment, the positive return from bump sensors means that Create touches the wall. . The bumper sensors return feedbacks only when they detect a hard surface. In our experiment, the positive return from bumper sensors means that robot touches the wall and then, high weight will be assigned to the particles which are around the wall, and low weight will be given to the rest of particles.

2.6 Summary

The robot localization problem is the problem of answering the question "*Where am I?*" from a robot's point of view. In some cases, the robot has access to priori information (map) that is describing characteristics of the environment. In other cases, however, the robot acquires the information while it is localizing in the environment. This information consists of relative and absolute measurements. The relative information provides high frequency, low cost, detailed information about the relative

displacement of the robot, independent of features in the environment [46, 43]. The absolute information provides position measurements based on observations made from the environment. This position information is independent of previous position estimates. However, this comes at the price of higher computational costs, lower frequency and lower accuracy. Since the absolute measurements do not depend on previous position estimates, they do not suffer from unbounded error growth.

In probabilistic localization problem, the robot considers the whole space of locations as a possible location to be, instead of being sure of one location. A robot starts with an initial belief and this belief can be a uniform distribution when the robot has no idea where it is, or it can be a distribution with one peak at the right location if the robot knows, or thinks it knows, where it is.

Chapter 3

Robot localization with known orientation

As we mention above, among many localization techniques, MCL has become arguably the most popular approach to date. However, the standard MCL technique sometimes is unable to maintain multimodal belief distributions that are present in complex situation such as symmetric environments [50]. We propose a new method based on this localization problem in symmetric environment and we will discuss more about it in this chapter.

In Section 3.1 we present our motivation based on the MCL's debilities. In Section, 3.2, we propose our method according to this motivation. Then we discuss clustering and explain how it works in Section 3.3. We discuss popular algorithm for clustering in section 3.4 and introduce the basic sequential algorithm as a fast method to produce a single clustering. Section 3.5 describe more details about our method and finally we may draw our conclusion in section 3.6

3.1 Motivation

Undoubtedly, many algorithms have been implemented based on MCL [19, 20, 36, 38] in recent years. In most of them, increasing the accuracy and efficiency of MCL are intended as a main goal. For example one of the controversial issues is when the number of particles required to achieve a certain level of accuracy varies drastically [3,47]. There are several extensions to MCL that solve the problem of failure in number of particles. Take for example, Sensor Resetting Localization (SRL), Mixture MCL (Mix-MCL), and Adaptive MCL (A-MCL) [22]. Although some of those approaches produce remarkable results, they are not satisfactory, due to the feature of the environments. This is especially true for any application in dynamic or symmetric environment such as

Robocop soccer field which is dynamic because other robots as a soccer players are moving through the environment and it is symmetric because objects are reflecting along the coordinate axis. This will increase the level of uncertainly for robot to find its actual position precisely when the environment is very ambiguous and symmetric. Therefore, not only method of implementation, but also feature of the environment plays a very important role to increasing the accuracy and efficiency of MCL.

However, our main concern in this thesis is based on the symmetric environment. We executed the ordinary MCL in this environment and the following three cases emerged.

First Case: Most of the generated particles are accumulated around the actual location of robot and confirmed that the localization is successfully ended. However, this is the case that does not happen very often in symmetric environment. Figure 3.1 illustrate the situation.

Figure 3.1: Particles are accumulated around robot

In this rectangle symmetric environment, Particles are accumulated in one location and robot is presented as a red circle in middle of that.

Second Case: In this case, particles have reflection along the coordinate axis. This reflection is achieved based on the movement of the robot in our environment. For example, when environment is rectangle and robot touches the long side of the environment the reflection will be based on the X-axis, because long side is parallel with x-axis. Same thing for small side, when robot touches the small side of our rectangular environment, the reflection will be based on the Y-axis because small side is parallel with y-axis. Figure 3.2 demonstrating this situation. As we can clearly see in this picture, when we divide our environment to four smaller areas along the coordinate axis, although robot is located in fourth section, particles are representing the virtual location of robot in second area. This is the situation when robot is localized vice versa.

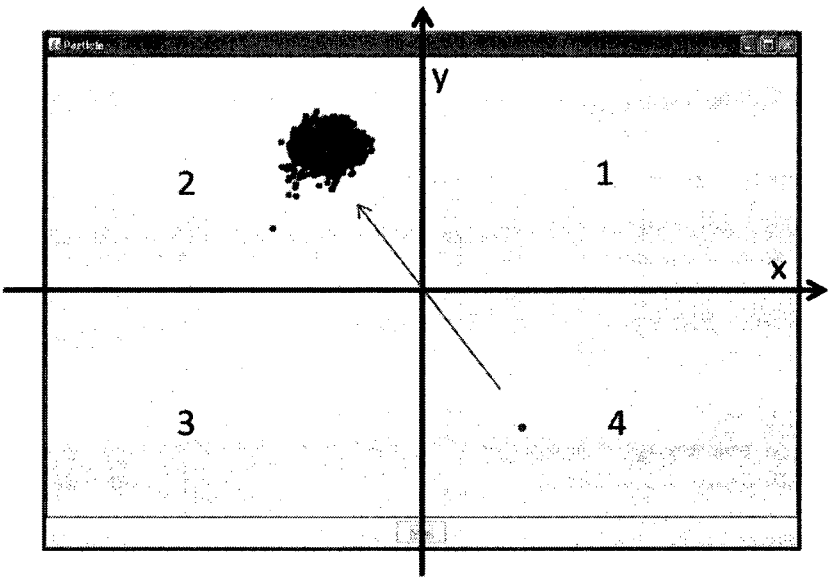


Figure 3.2 Robot is located vice versa

Third Case: this case which is the most common situation in symmetric environment is combination of two previous cases. This time, particles will show the location of robot not only in actual position, but also as a reflection along the coordinate axis. As we can see in figure 3.3, particles are

accumulated in two main group and illustrating the location of robot in second and forth area. Particles with black color represent the true location of robot and particles that are blue represent the virtual location of our robot.

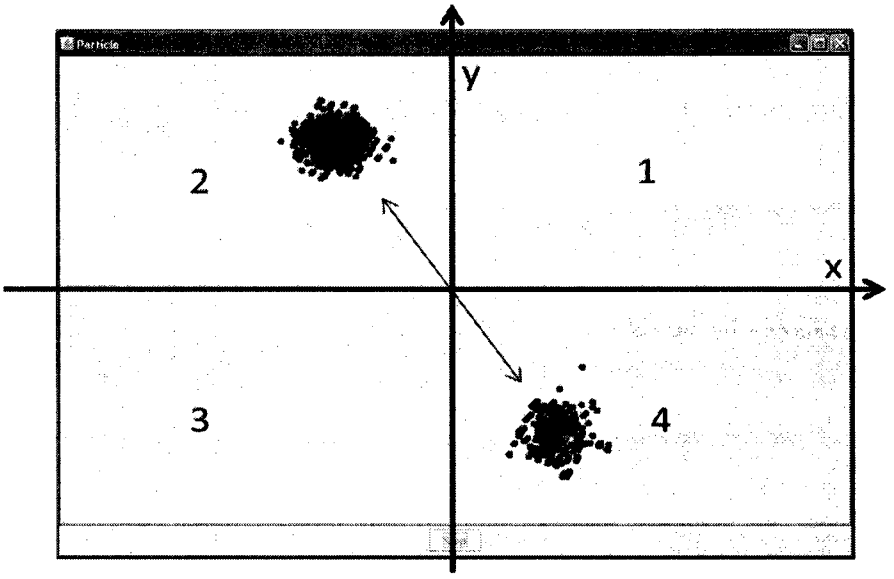


Figure 3.3 Combination of two previous cases and represents the location of robot in two areas.

3.2 Proposed Method

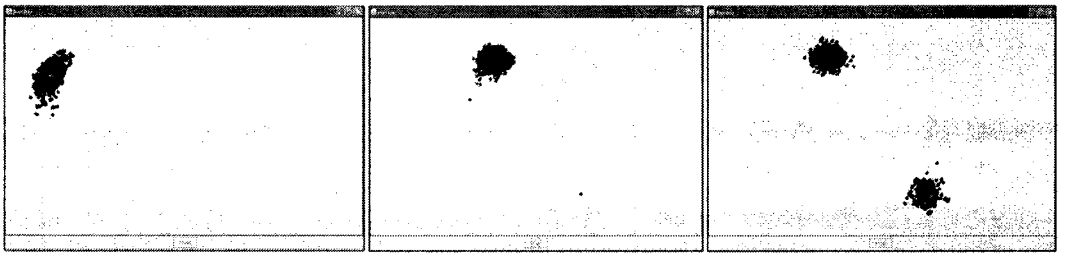
Before describing our method in details, we first explain the problem statement that gave us the motivation to propose our method in symmetric environment and then show how we solve this significant problem in our method.

3.2.1 Problem Statement

In robotic, when we talk about the localization, several factors must be considered. Method of implementation, environment and even accuracy of the sensors are just few of them. Each of them has its own hardness and consider as separate field. However, in case of environment, symmetric environment is one of the most challenging environments in robot localization. We consider our main focus in this environment and propose a method for mobile robot localization in symmetric environment. However,

because our method is based on Monte Carlo Localization framework we first executed the ordinary MCL in this environment. After iterating several steps in MCL, we realize that when samples are generated according to the posterior distribution (as is the case in MCL), they may represent the multimodal distributions that often arise during the localization in symmetric environment. We also find that in some cases particles are ignoring the possibility that the robot might be located in somewhere else and they often too quickly converge to a single, high likelihood pose regardless to the actual location of robot.

According to these inabilities in MCL, we propose a method to determine the actual position of robot in symmetric environment. In our method, we utilized the clustering because particles are not considered individually as a single point and we need to analyze them as a whole entity. Based on checking the orientation of each clusters, the robot can distinguish that which one of the above described cases is happed. Figure 3.4 shows three pictures of localization cases in symmetric environment.



(a) (b) (c)

Figure 3.4: An example of three stages of MCL in symmetric environment.

Figure 3.4(a) it shows particles concentrated successfully around the true position of the robot and the true position is represented by black cluster. In figure 3.4(b), although the robot is located in bottom side, particles are representing the virtual location of robot in opposite side. This is the situation when particles are accumulated in blue cluster and robot is localized vice versa. Figure 3.4(c) shows that there is uncertainty that whether robot is located in black or blue cluster.

3.2.2 Details of our method

To obtain a better result from MCL in symmetric environment, the distribution of the particle set in our method is analyzed by sending to the clustering part. Then, the resultant clusters are used to determine whether the robot is successfully localized in true location or it is localized contrariwise. In case of reverse localization, particles have the opposite direction with robot. For example, in figure3.5, when the value of the robot's orientation is equal to π ($\theta = \pi$) and robot is moving to the north side, particles return their orientation value equal to 2π and moving to the south side.

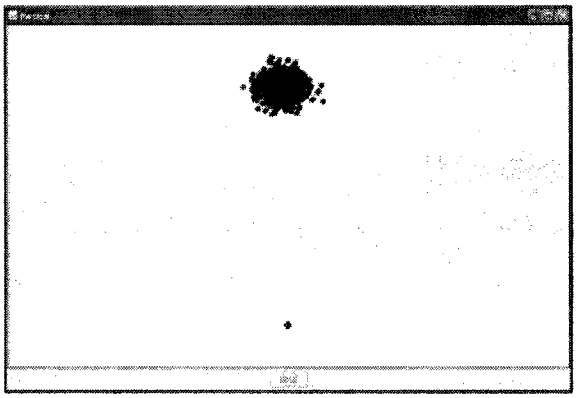


Figure 3.5: Case of reverse localization

Therefore orientation is considered as a main point in our method to distinguish about which cluster is correspond to direction of the robot and which cluster is pointed to another bearing. In order to describe our consequence in this method, two characteristic variables are calculated respectively as below:

1) θ, the value of representative point (robot's orientation).
2) Θ, threshold of dissimilarity

θ is indicate as a value of our robot's orientation and it is initially postulated parallel with the direction of *x-axis* ($\theta = \pi$) in first time that we settle our robot in our environment. This value is updated whenever that robot is turned in our environment. Θ which is postulated to 20° was used as dissimilarity input value to measuring the dissimilarity

between current particle and every existent cluster in our environment. The value of Θ is derived from the experiment. We have appointed different values for Θ during our experiment but we realized that the best acceptable value for alpha is 20 degrees considering to the size of our environment and the percentage of the error distance in our motion model. However, if the value of this measurement was larger or smaller than Θ, a new cluster that contains current particles will be created otherwise, the considered particle will be assigned to the existing cluster which has a minimum dissimilarity measure to it. However, before we are going to describe our method in more details, a brief introduction will be provided to introduce the clustering and shows how it is works in our algorithm.

3.3 Clustering Algorithm

Clustering is one solution to the case of unsupervised learning, where class labeling information of the data is not available. Clustering is a method where data is divided into groups (clusters) which 'seem' to make sense. Clustering algorithms are usually fast and quite simple. They need no beforehand knowledge of the used data and form a solution by comparing the given samples to each other and to the clustering criterion. Clustering is used in many fields of science including machine vision, life and medical sciences and information science. One reason for this is the fact that intelligent beings, humans included, are known to use the idea of clustering in many brain functions.

3.3.1 Basic Concept

When classifying different kind of samples a way to represent the sample in a mathematical way is needed. These features are represented in a feature vector. A feature vector is a vector including different features for the sample. That is, with n features x_i the feature vector is of the form

$$x = [x_1, x_2, \ldots, x_n]^T \qquad (3.1)$$

Where T denotes transposition and x_i is typically real numbers. The selection of these features is often very hard due to the fact there usually are a lot of features from where the most representative ones should be selected. This is because the computational complexity of the classification (clustering) algorithm grows with every feature selected. Feature selection and the reduction of dimensionality of the data are beyond this document.

3.3.2 Definition of a Cluster

Now, let us define some basic concepts of clusters in a mathematical way. Let X be a set of data, that is

$$X = [x_1, x_2, \ldots, x_n] \qquad (3.2)$$

Where X is a set of vectors constituted by n vectors x_i. The set X includes a group of vectors, into m small sets (clusters C_i) if the following conditions are met:

1. None of the clusters is empty; $C_i \neq \emptyset$

2. Every sample belongs to a cluster

3. Every sample belongs to a single cluster $C_i \cap C_j = \emptyset$, $i \neq j$

Naturally, it is assumed that vectors in cluster C_i are in some way "more similar" to each other than to the vectors in other clusters.

3.3.3 Proximity Measure

When clustering is applied a way to measure the similarities and dissimilarities between the samples is needed. A typical case where proximity between subsets is needed is when a single vector x is measured against a cluster C. The representative can be chosen so that the value is, for example, maximized or minimized. If a single vector representative is chosen among C the used method is called global clustering criteria

and if all the vectors in C have an effect on the representative a local clustering criteria is being used. Figure 3.5 one type of representatives is shown below.

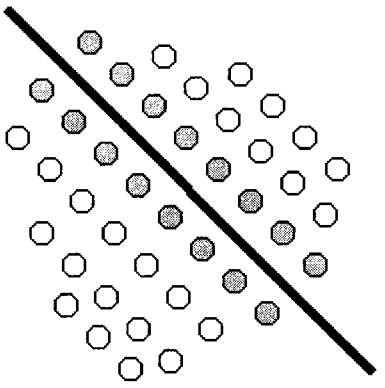


Figure 3.6: Line representative [17]

3.4 Popular algorithm in Clustering

As we mention above, calculating all possible combinations of the feature vectors is not generally possible. Clustering algorithms provide means to make a sensible division into small clusters by using only a fraction of the work needed to calculate all possible combinations.

3.4.1 Basic Sequential Algorithm

Sequential algorithms are straightforward and fast methods to produce a single clustering [32]. Usually the feature vectors are presented to the algorithm once or a few times. Final result is typically dependent on the order of presentation and the result is often compact.

A very basic clustering algorithm that is easy to understand is basic sequential algorithmic scheme (BSAS). In the basic form vectors are presented only once and the number of clusters is not known a priori. What is needed is the dissimilarity

measure d(x, C) and the threshold of dissimilarity Θ and the number of maximum clusters allowed q.

The idea is to assign every newly presented vector to an existing cluster or create a new cluster for this sample, depending on the distance to the already defined clusters. As pseudo the algorithm works like in table 3.1.

1. $m = 1; C_m = \{x_1\};$ // Init first cluster = first sample
2. for every sample *x* from *2* to *N*
   a. find cluster C_k such that min $d(x, C_k)$
   b. if $d(x, C_k) > \Theta$ AND $(m < q)$
      i. $m = m + 1; C_m = \{x\}$ // Create a new cluster
   c. else
      i. $C_k = C_k + \{x\}$ // Add sample to the nearest cluster
      ii. Update representative if needed
3. end algorithm

Table 3.1 Basic Sequential Algorithm

As can be seen the algorithm is simple but still quite efficient. Different choices for the distance function lead to different results and unfortunately the order in which the samples are presented can also have a great effect to the final result. What's also very important is a correct value for Θ. This value has a direct effect on the number of formed clusters. If Θ is too small unnecessary clusters are created and if too large a value is chosen less than required number of clusters are formed.

One detail is that if q is not defined the algorithm 'decides' the number of clusters on its own. This might be wanted under some circumstances but when dealing with limited resources a limited q is usually chosen.

3.5 Further details for our method

The concept of localization in our method is referred as the different distributions of particle set which have significant characteristics and can be distinguished from each other. In this method, the distribution of the particle set is analyzed by sending to the clustering part and then, the clustered particle set is further used to extract information. This characterization is based on the orientation and particles will be organized according to this attitude. Therefore the orientation of each individual particles has been compared with orientation of our robot that is initially postulated parallel with the direction of the *x-axis* ($\theta = \pi$). For this comparison, threshold Θ is considered as an input and it is equal with 20°. Based on this threshold value, we can estimate that the maximum number of cluster that may produce is equal to18 clusters. However, most of these clusters will disappear after iterating several steps in MCL and only two of them will remain in our environment frequently. One of these clusters always refers to actual location of robot with same orientation and other one always acts vice versa. Just to have a better visual perception, we have shown each one of these clusters with different color that makes our comparison easier. For example, particles in black cluster always represent the same orientation with robot and particles in blue cluster act contrary. Furthermore, in order to describe the stages of localization, three characteristic variables are calculated respectively. n_c for number of clusters, n_{max} for number of particles in the cluster which has the maximum number of particles compared to other clusters, and p_{max} for percentage of n_{max} in the current whole set of particles. We utilized these parameters to have a terminate condition to stop the recursive process in MCL. For example, if one of our clusters contained more than 80% of particles in whole particle set, the localization process will stop. In case of having two clusters such as case three in our environment, when number of particles in both cluster become more than

90% of particles in whole particle set, the robot will stop to indicating the localization process.

We also specified a yellow point in middle of the biggest cluster - in case of the number of particles - to represent the actual location of our robot. In order to do that, we choose Euclidean distance between particles as our proximity measures. However, if this point is appeared in middle of the black cluster, the distance between the current location of robot and actual location of yellow point is not very significant. On contrary case, when the yellow point is appeared in middle of the blue cluster, we will represent that backward and reflect it along a coordinate axis in our environment. In this case the distance between the current location of robot and actual location of yellow point is significant. Table 3.1 is pseudo code description of our method that is represented as below.

The combined MCL-Clustering algorithm

$X_k = \text{MCL}\ (x_{k-1}\ , u_k\ , z_k\ )$

$C_k = \text{BSAS}\ (x_k\ , \Theta)$ //clustering based on Orientation

//number of clusters in clustered particle set

$n_c\ = \text{numberOfClusters}(C_k)$

//the number of particles in the cluster which has the maximum number of particles compared to other clusters

$n_{max} = \text{maxParticleNumbers}\ (C_k)$

//the percentage of n_{max} in the current whole set of particles, n_{total} is the number of current particles

if $(C_o = \theta)$ & $(\ p_{max} > e)$

//stop to indicate localization is successful completed or starts doing other jobs

Return $X_k\ , C_k\ , n_c\ , n_{max}\ , p_{max}$

Table 3.2: Incorporated MCL with clustering algorithm

3.6 Summary

We mixed the clustering method with MCL framework to categorize this information and have a better resolution result. Clustering is one solution to the case of unsupervised learning, where class labeling information of the data is not available[41]. Clustering is a method where data is divided into groups (clusters) which 'seem' to make sense. One of the fastest ways to produce a clustering is basic sequential algorithmic scheme that we employed it to do our clustering which is based on orientation.

In our method, the orientation of each individual particles has been compared with orientation of our robot that is initially postulated parallel with the direction of the *x-axis* .Therefore, those particles with same orientation range are collected in the same cluster and the rest will goes and collected in other cluster. Then we can distinguish about which cluster is correspond to the direction of the robot and which cluster is pointed to another bearing. This technique has a noticeable functionality in symmetric environment and will improve the ambiguous of belief state.

Chapter4

Implementation and Experiment Results

In this chapter, we show how we can apply our method to the Robot Localization problem in symmetric environment. We implemented a simulator and real robot word experiments that allows us to step-by-step combine a part of the localization problem with our technique and look at how the behaviors of our method in different circumstances, considering practical situations.

In Section 4.1 we start with some general remarks about preparation, Robot Serial Command Interface (SCI) and its behavior. In section 4.2 hardware platforms and its setup has been discussed. In section 5.3 we will present the experimental results and then provide some references to related work in Section 4.4.

4.1 Preparation

In order to implement our method in virtual and real word environment, we prepared of list of hardware and software interface that we described each one of them in details respectively.

4.1.1 Software Interface

Versions of Robot manufactured contain an electronic and software interface that allows us to control or modify our robot behavior and remotely monitor its sensors. Our robot is not exemption and utilized the interface that called the *iRobot Create Serial Command Interface* or *Create SCI*.

Create SCI is a serial protocol that allows us to control our robot that called *Create* which is an autonomous mobile robot for educators and developers built by iRobot Corporation, through its external serial port (Mini-DIN connector). The SCI includes commands to control all of Create's actuators and also to request sensor data from all of Create's sensors. Using the SCI, we can add functionality to the normal Create behavior or we can create completely new operating instructions for Create.

To use the SCI, a processor capable of generating serial commands such as a PC or a microcontroller must be connected to the external Mini-DIN connector on Create. The connector is located in the rear right side of Create beneath a snap-away plastic guard.

4.1.2 Create SCI Pattern

The Create SCI has four operating modes: off, passive, safe, and full. On a battery change or other loss of power, the SCI will be turned off. When it is off, the SCI will listen at the default baud bps for an SCI Start command. Once it receives the Start command, the SCI will be enabled in passive mode. In passive mode, we can request and receive sensor data using the Sensors command, execute virtual button pushes to start and stop the cycles and define a song (but not play one).

We cannot control any of Create's actuators when in passive mode, but Create will continue to behave normally, including performing, charging, etc. When in passive mode, we can then send the Control command to put the robot into safe mode. In safe mode, we have full control of the robot, except for detection of a cliff while moving forward (or moving backward with a small turning radius), detection of wheel drop (on any wheel) and charger plugged in and powered.

When one of the conditions listed above occurs, the robot stops all motors and reverts to passive mode. For complete control of the robot, we must send the Full command while in safe mode to put the SCI into full mode. Full mode shuts off the cliff and wheel-drop safety features. (The robot will still not run with a powered charger

plugged in.) This mode gives us unrestricted control of the robot's actuators. To put the SCI back into safe mode, we can send the Safe command.

If no commands are sent to the SCI when it is in safe or full mode, Create will wait with all motors off and will not respond to button presses or other sensor input. To go back to passive mode from safe or full mode, we can send any one of the four virtual button commands. These button commands are equivalent to the corresponding button press in normal Create behavior.

However, all the Create's controls such as movement and access to sensors are obtained through a Java *Application Programming Interface* (API) named Roombacomm, which is Java library for communicating and controlling the Create. Although it is designed for robot Roomba, it works very well with Create and it works on any operating system that supports a serial communicator for Java (RXTX). Therefore, all source code in our implementation is written in Java with utilizing the Eclipse Software Development Kit.

4.1.3 Hardware Platforms

The Roomba is an autonomous robotic vacuum cleaner made and sold by iRobot. Under normal operating conditions, it is able to navigate a living space and its obstacles while vacuuming the floor. However, to give scientists a better platform they've gone ahead and built the *iRobot Create* to work with. The main differences are a lack of vacuum - no more clean floors - and a nifty "cargo bay connector" which can support a Command Module which bristles with ports and allows us-added motors, sensors and the like. Otherwise, most functions are quite similar to that such as Virtual Walls, the Home base and the Remote Control.

The iRobot Create comes fully assembled. It has two powered wheels, a castor (and optional 4th wheel), 10 pre-programmed behaviors, an expandable input/output port for custom sensors and actuators a cargo bay with mounting points and a tailgate for ballast. As a first movement, because the Create is designed to move forward, all the

sensitive sensors are located on the movable front bumper. This rubber bumper protects them or anything they run into from any damage that might otherwise be sustained. When a Roomba turns, its fixed front wheel would skid. This swiveling caster of the Create reduces that and hopefully makes turning a little more accurate.

The Cargo Bay Connector, located in the front middle of the cargo bay, contains 25 pins that let us to attach electronics for peripheral devices such as additional sensors. The Element Direct BAM (short for Bluetooth Adapter Module) is one of these additional sensors that enable wireless control of the iRobot Create robot from a Windows. The BAM connects to the Create's cargo bay port – without any extra wires or cables. The BAM provides a virtual serial port connection between a Bluetooth host and Create. A PC can communicate with Create in the same way it would as if it were attached with a serial cable. The BAM gives us complete wireless control of Create. It also exposes Create's programmable IO, making it easy to connect additional hardware.

Generally speaking, the iRobot Create is a great and inexpensive robotics platform, especially when compared to similar platforms aimed at academia. It uses standard Roomba parts for many of its subsystems, making it cheap to repair. The new commands and capabilities can lead to some interesting experiments with minimal added hardware.

Although the cheaper solutions exist for those on a budget, the Command Module is a good device for those desiring a quick and highly-integrated way to add intelligence to the Create.

4.2 Implementations of our Method

The performance of our method is tested on both real and simulated robots environment. The goal of the experiments is to verify and solve the localization problem in symmetric environment according to the distribution of the particles in our clusters. However, in both of these environments, we considered not only those three

important characteristic variables, but also designated another one called yellow point that we will explain it below.

As we discussed in previous chapter, we start comparing all the existing particles with our robot in base of the orientation and organized them in the way that they suppose to be. Two clusters have been considered that first one with black color, is corresponding to the true orientation of the robot and therefore demonstration the actual location of our robot and the second one with blue representation, is corresponding directly to opposite orientation that illustrating the position of our robot vice versa. For the rest of our particles, we will generate the new clusters, depends on the orientation on each particle, and collecting them on related particle set and then represent them with different color.

4.3 Experimental Results

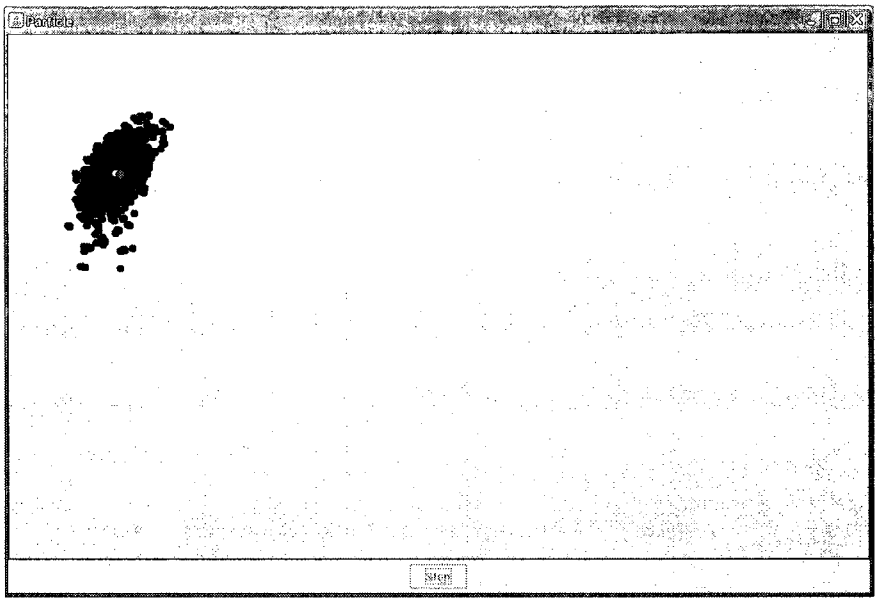
As we mention in section 3.2.1, three conditions have been considered in our method. However, all these three condition are utilized in both real and simulated environment. Below we explain each one of the in more details.

4.3.1 Simulation Environment

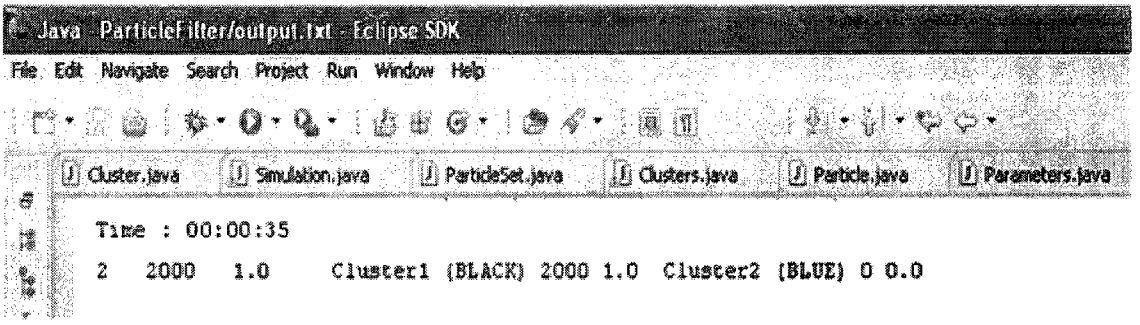
We placed our robot in a field of 900x550pixel and considering two thousand particles in this field to estimating the true location of our robot. These particles are randomly generated all over the environment and each one representing a different x, y, and θ.

Case one: The robot moves with known initial orientation ($\theta = \pi$). Based on this initialization, we execute our algorithm and as a first condition, robot and particles are become in the same location and robot is sitting in middle of the black cluster. In this case (Figure 4.2) particles in black cluster contained more than 80% of particles in whole particle set and therefore the localization process will stop Furthermore, the value of error distance is not very significant in this case because our

robot is located in black cluster which is means all the particles in this cluster have the same orientation as robot has.



(a)

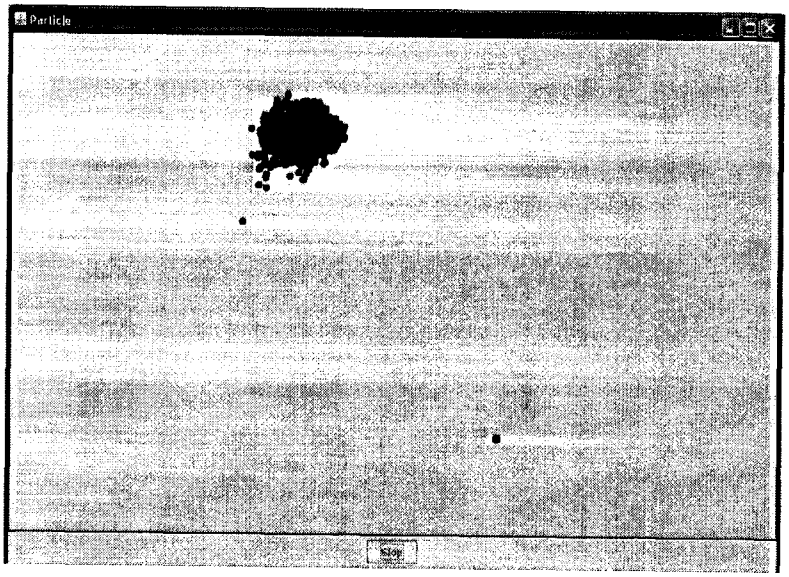


(b)

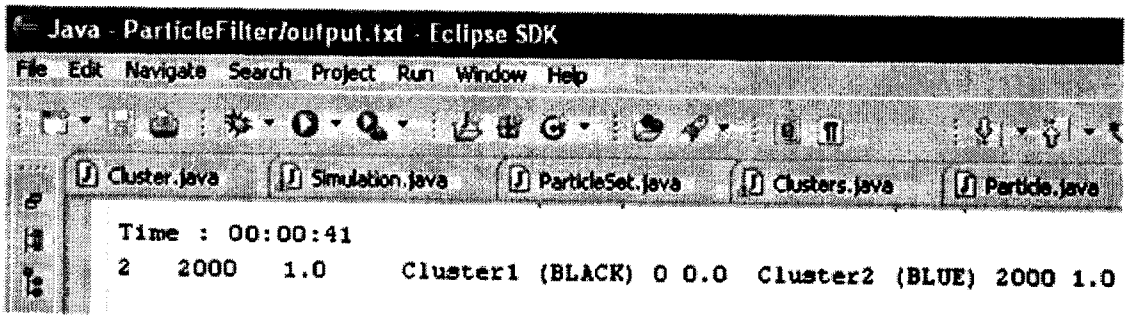
Figure 4.1: Robot is successfully localized

Figure 4.1 (a) shows the distribution of the particle in black cluster and current location of our robot in middle of that. The yellow point is representing the actual position of our robot in center of black cluster. Figure 4.1 (b), demonstrate that, most particles are accumulating in beigest cluster which is the black cluster and blue cluster is out of any particle.

Case Two: In next condition, all Particles are accumulated in blue cluster and therefore representing the opposite location of our robot. However, in this case, our robot is still in black cluster which is out of any particles. This time, because we already know the blue cluster is always showing the opposite location of our robot, the center of converse position of this cluster represents the actual position of our robot. Therefore, the value of the error distance in this case is very significant. Figure 4.3 shows the discussed condition below.



(a)

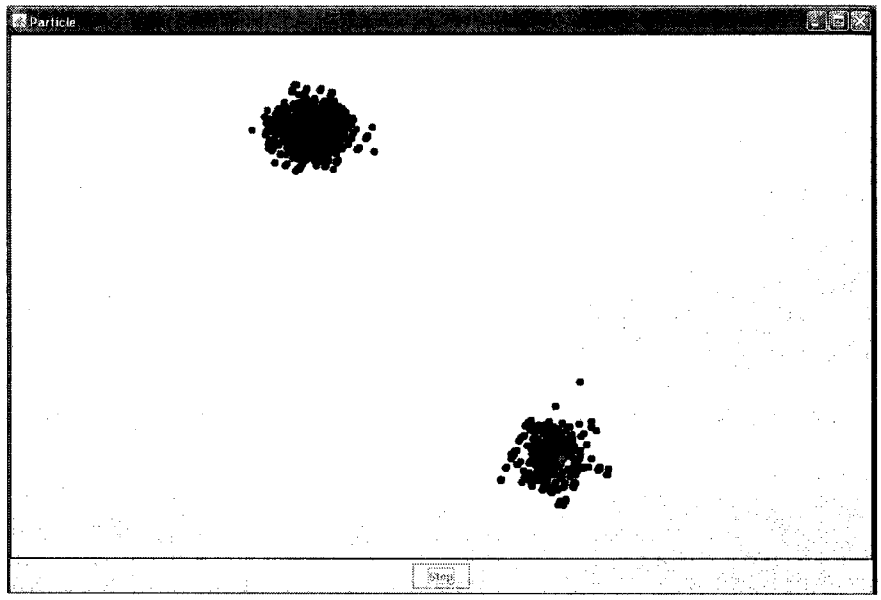


(b)

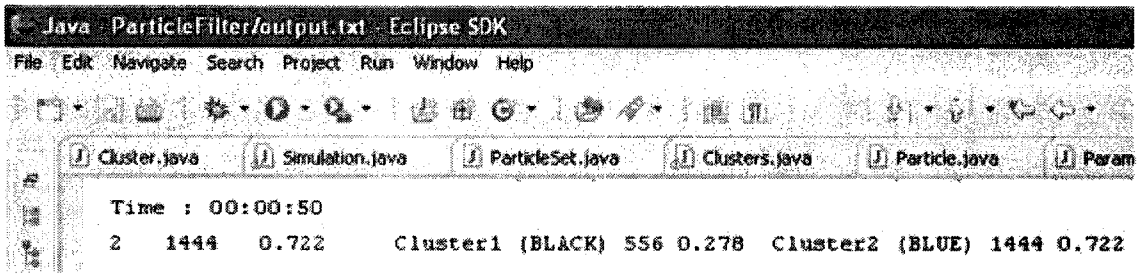
Figure 4.2: Robot is localized vice versa

Figure 4.2 (a) shows the distribution of the particle in blue cluster and current location of our robot which is in opposite side. In this case, the yellow point is represented the actual position of our robot based on the blue cluster. Figure 4.2 (b), demonstrate that all two thousand particles are accumulated in blue cluster which is the biggest cluster in this case and black cluster is out of any particle. Same as other situation, because more than 80% of particles in whole particle set are accumulating in one place, the localization process will stop although they represent the location of our robot contrariwise.

Case Three: In this condition we have two particles set with different orientation and it will be the most difficult situation to verify which one shows the robot true pose. Furthermore, In this case none of these clusters are contained more than 80% of current number of total particles. Therefore, stop condition is not considered and then algorithm will run for ever. To overtake of this problem we offer the solution that if total amounts of particles in both cluster – blue and black - become more than 90% the localization has been done and the algorithms will stop running. However, if particles that accumulated on black cluster are more than blue one, the error distance is not really noticeable. Instead, when particles are accumulated in blue cluster, the same step as case two will happen again and therefore, the value of the error distance will be very significant. Figure 4.3 shows the discussed condition below.



(a)

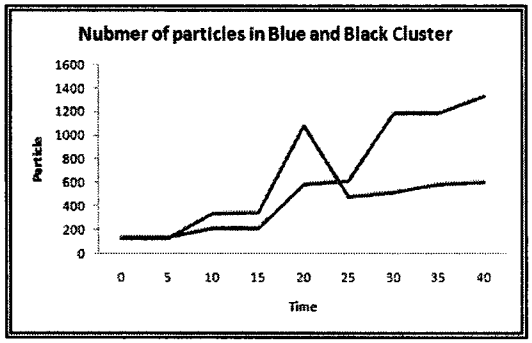


(b)

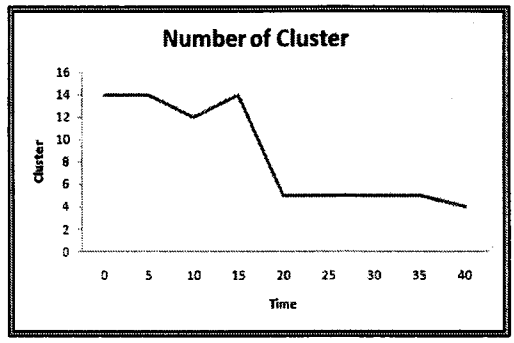
Figure 4.3: Localization in symmetric environment

Figure 4.3 (a) shows the distribution of the particle in both blue and black cluster and current location of our robot in middle of black cluster. In this case, the yellow point is represented the actual position of our robot depends on each cluster that consist more particle than other. For black cluster, case one is considered and for blue cluster case two is considered. Figure 4.3 (b), demonstrate the number of cluster in first column (n_c = 2), number of particles in beigest cluster in second column (n_{max} = 1444) and percentage on

n_{max} in third column (p_{max} = 0.722). It also shows all two thousand particles are divided between black and blue cluster



(a)



(b)

Figure 4.4: The plots of corresponding (a) Number of particles in blue and black cluster (b) number of clusters.

Time	Number of Cluster	Number of particle in bigest cluster	Percentage of particle in bigest cluster	Number of particle in Black Cluster	Percentage of particle in Black cluster	Number of particle in Blue Cluster	Percentage of particle in Blue cluster	Number of particle in Black and Blue cluster	Percentage of particle in Black and Blue cluster
0	14	172	0.086	134	0.067	124	0.062	258	0.129
5	14	179	0.0895	132	0.066	121	0.0605	253	0.1265
10	12	590	0.295	210	0.105	335	0.1675	545	0.2725
15	14	339	0.1695	210	0.105	339	0.1695	549	0.2745
20	5	1087	0.5436	586	0.293	1087	0.5435	1673	0.8365
25	5	617	0.3085	617	0.3085	474	0.237	1091	0.5455
30	5	1192	0.596	1192	0.596	519	0.2955	1711	0.8915
35	5	1192	0.596	1192	0.596	587	0.2935	1779	0.8895
40	4	1337	0.6685	1337	0.6685	607	0.3035	1944	0.972

Table 4.1: The value of all parameters in above simulation result

Figure 4.4(a), show the accumulation of particles in both cluster. It also demonstrates the number of particles growth more rapidly in blue cluster after 15 second. Figure 4.4(b), illustrate that the number of cluster decreased rapidly after 15 second and then remain steady up to time 40.

Table 4.1 represents the value of all parameters in our process. First column shows the time value that we spend in whole process. The second column shows the numbers of clusters fall from 14 to 4. Next two columns illustrate the information about

biggest cluster. Information in fifth and sixth column is about black cluster. As we can see, after 30 second the black cluster becomes the biggest cluster in our environment. Number and percentage of the particles in blue cluster is demonstrated in next two columns. This table also shows the number of particles in both cluster and percentage of the relevant in next two columns.

4.3.2 Real Environment

In this section, we present the experimental results performed on our real robot that obviating of localization problem in symmetric environment. As a result, we will see that our robot is going to stop when a large number of dispersed particles are accumulating and also will successfully localize when the particles are located on one or both of our blue and black cluster. It also returns the true location and orientation according to the clusters and the yellow point measurement that we explain that in previous section.

We executed our method, starting by placing our robot in a field of $100_{cm} \times 150_{cm}$ around with wall and because we initialized or orientation before ($\pi = 180°$), we adjust face of our robot parallel with X axis. Figure 4.6 shows how we settle our robot in our symmetric environment. In this experiment, the number of particles is initialized as 2000 particles and the criterion for clustering is $\Theta = 20°$. Criterion Θ is a threshold used in BSAS to determine whether a particle belongs to an existing cluster or is assigned to a newly created cluster.

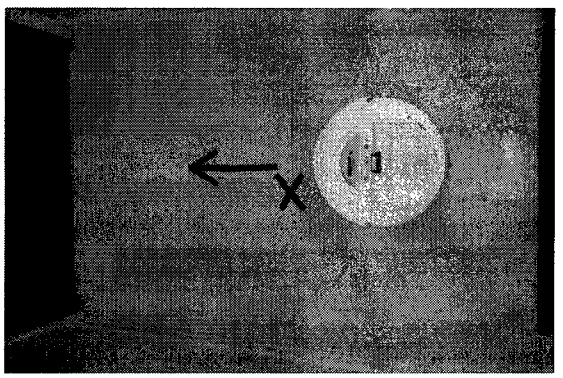
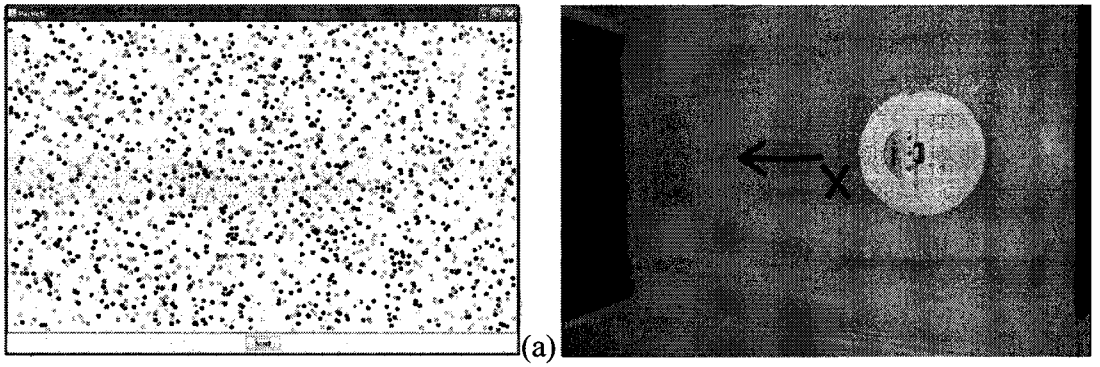
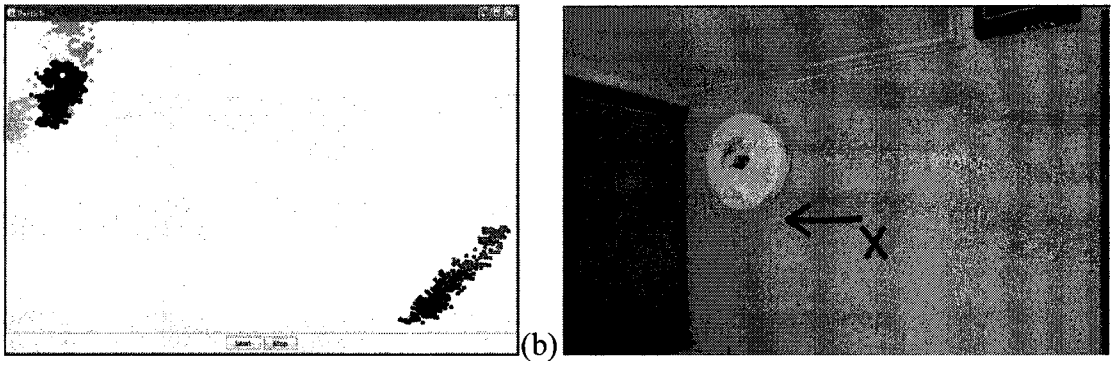
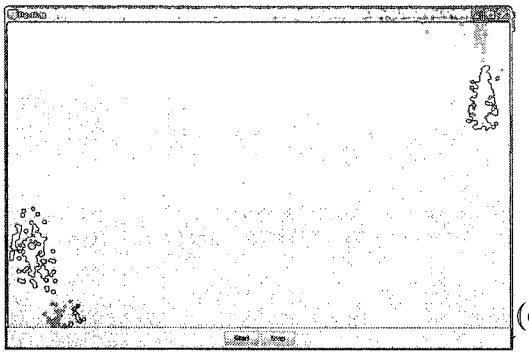


Figure 4.5: Face of our robot is paralleled with X axis

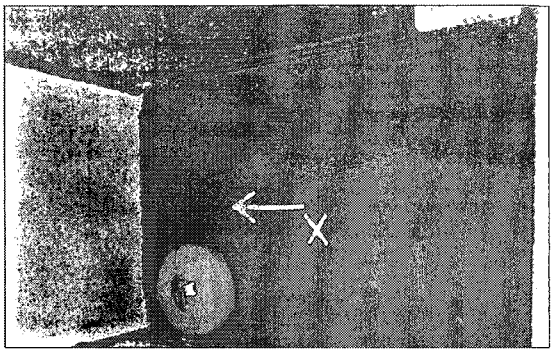
We also considering two more parameters that indicate the robot is localized or lost. First parameter is considered as 20% of total particles, which means when the number of particles in the largest cluster is lower than 20% of whole particles, the robot will believe it has lost. For second parameter, we postulate 80% of whole particle, which means when the number of particles in largest cluster is equal or more than 80% of whole particles, the robot will believe it is localized successfully. However, in case three, because particles are separated in both clusters, we postulate our second parameter as a 90% of whole particles, in both blue and black cluster to show that our robot is successfully localized or not.

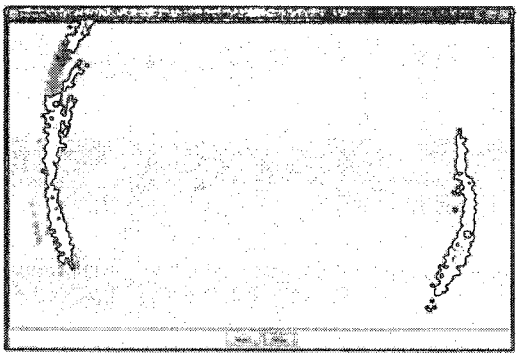




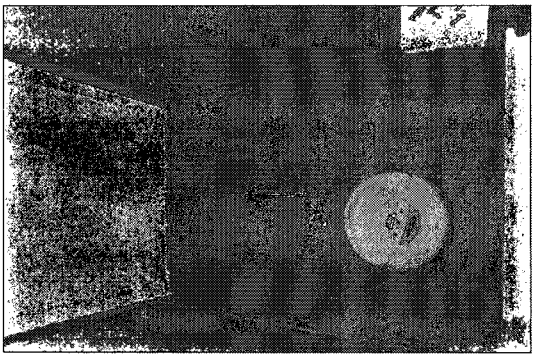


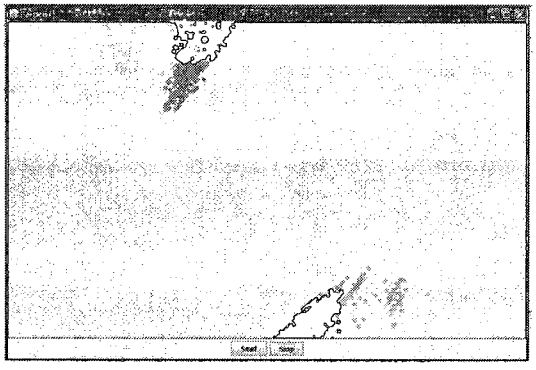
(c)



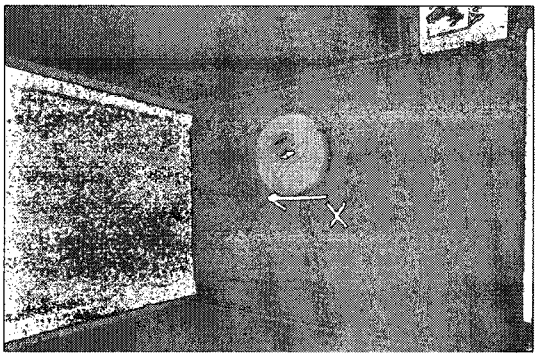


(d)





(e)



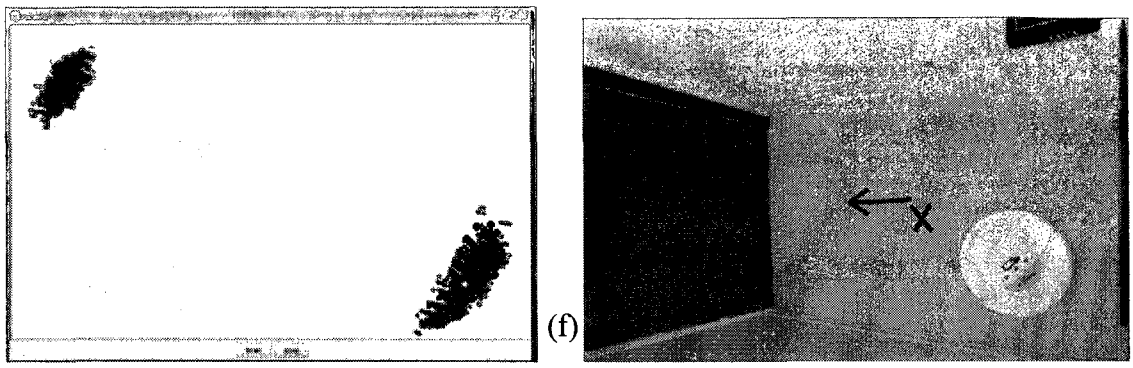


Figure 4.6: The robot true pose and distribution of particles during our experiment

Figure 4.6 illustrate our experiment in real word. Figure 4.6(a) shows that particles are located all over our environment. Because we randomly generate them, each one has different orientation and therefore each one will represent with different color. Figure4.6 (b) shows that we have 6 clusters but none of them have more that 80% of whole particles, therefore our process will keep running and goes to next figure. In Figure 4.6(c), although our robot moves to different location, we still have more than 2 clusters with less than of 80% of whole particles. This process is continuing until last step in figure 4.6(f), that we have only two clusters. Black cluster is then representing the true location of our robot and the yellow point in middle of that shows the actual location of our robot.

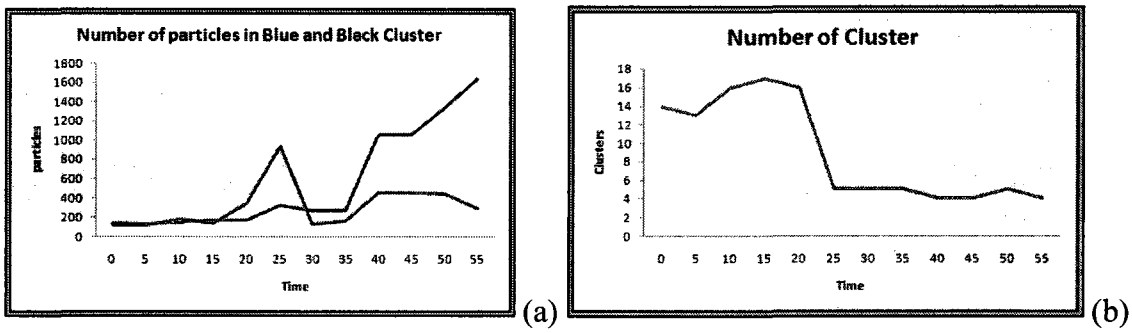


Figure 4.7: The plots of corresponding (a) Number of particles in blue and black cluster (b) number of clusters.

Figure 4.7(a) indicates that the number of particles in blue and black cluster and 4.7(b) shows how the number of cluster is decreased from 18 to 4 clusters between time 15 and 55 second.

Time	Number of Cluster	Number of particle in bigest cluster	Percentage of particle in bigest cluster	Number of particle in Black Cluster	Percentage of particle in Black cluster	Number of particle in Blue Cluster	Percentage of particle in Blue cluster	Number of particle in Black and Blue cluster	Percentage of particle in Black and Blue cluster
0	14	184	0.092	142	0.071	116	0.058	258	0.129
5	13	185	0.0925	126	0.063	113	0.0565	239	0.1195
10	16	339	0.1995	147	0.0735	183	0.0915	330	0.165
15	17	397	0.1985	169	0.0845	136	0.068	305	0.1525
20	16	389	0.1945	166	0.083	336	0.168	502	0.251
25	5	938	0.469	322	0.161	938	0.469	1260	0.63
30	5	1213	0.6065	265	0.1325	129	0.0645	394	0.197
35	5	1182	0.591	266	0.133	161	0.0805	427	0.2135
40	4	1050	0.525	1050	0.525	449	0.2245	1499	0.7495
45	4	1049	0.5245	1049	0.5245	451	0.2255	1500	0.75
50	5	1334	0.667	1334	0.667	441	0.2205	1775	0.8875
55	4	1635	0.8175	1635	0.8175	282	0.141	1917	0.9585

Table 4.2: The value of all parameters in above experiment result

Table 4.2 represents the value of all parameters in our real experiment. First column shows that we spend 55 second to obtain the suitable result in our whole process. Second column shows the numbers of clusters. This number rapidly decreased from 16 to 5 after 20 second. Next two columns illustrate the information about biggest cluster. This number is gradually raised after 30 second. Information in fifth and sixth column is about black cluster. As we can see, after 40 second the black cluster becomes the biggest cluster in our environment. Number and percentage of the particles in blue cluster is demonstrated in next two columns. This value has been very fluctuated. This table also shows the number of particles in both cluster and percentage of the relevant in next two columns.

4.4 Limitation of our Method

Our framework is based on Monte Carlo Localization, which draws samples uniformly at random from the environment free-space. This process has some

disadvantages for localization. Take for example the process can demand a high number of particles to completely cover the environment in order to guarantee that the robot will be able to recover its pose. It is known that the performance of the Monte Carlo filter highly depends on having some particles with a pose close to the real robot pose in the initial distribution. Due to this inherent fault, MCL may fail during localization. Therefore, we have suggested to make sure the failure of MCL will not occur when using our method to help robot know whether it is successfully localized in your environment.

Chapter 5

Conclusion and Future Work

In this work we have thoroughly discussed the problem of robot localization in symmetric environment and then show how to apply our techniques to solve this problem. We have pointed out advantages and disadvantages of our technique and we have discussed the use of our method for all possible cases in symmetric environment, illustrated with experiments.

5.1 Conclusion

We proposed a new method to improve the localization problem in symmetric environment. This method which is based on MCL framework has been executed in both real and simulated environment. In case of localization, most existing approaches focus on the accuracy and efficiency of MCL by adding more and more particles until better observation likelihoods can be obtained.

However, one drawback is the inability to deal with local maxima that are present in symmetric environments. In this thesis we proposed an algorithm that mainly focus on this problem and help robot to successfully localize itself in symmetric environment. In order to do that, we initialize the orientation of our robot, and utilized the basic sequential algorithm for clustering the particle set in real time. This aggregation which is based on orientation will help us to distinguish the right particle set (cluster) that present the true location of robot. Beside of that, by considering the number of clusters and the number of particles in each one of them in our particle set, we realize whether if robot is successfully localized or not.

5.2 Future Work

This work can be used for further studies in a number of different directions.

Kidnapped problem: according to our initialization, it is possible to verify the failure case in kidnapped problem and recover it aging after transported our robot to some other place without being told.

Resampling: Our method is based on MCL framework. So it has the inherent limitation of MCL which is particle deprivation problem. In some cases, even with a large number of particles, it may happen that there are no particles around the correct state. For future work, we may generate new particles based on their weight and their orientation. and then do more measurements to verify if the state showed by particles is correct.

Accelerating: this method is proposed in case of verifying the localization, not aimed at speeding up the robot localization. Therefore one objective for future researcher is to control the robot so as to minimize the speed of localization in our method.

References

[1] J. Borenstein, B. Everett, and L. Feng. Navigating Mobile Robots: Systems and Techniques. A. K. Peters, Ltd., Wellesley, MA, 1996.

[2] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. Journal of Artificial Intelligence Research, (1999), 391–427. Available at: http://www.jair.org/papers/paper616.html

[3] B. Yamauchi and R. Beer. Spatial learning for navigation in dynamic environments. IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, Special Issue on Learning Autonomous Robots, 1996. Available at: http://www.nrl.navy.mil/aic/index.php

[4] B. Schiele and J. Crowley. A comparison of position estimation techniques using occupancy grids. In Proceedings of the 1994 IEEE International Conference on Robotics and Automation, pages 1628–1634, San Diego, CA, May 1994.

[5] G. Wei, C. Wetzler, and E. von Puttkamer. Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans. In Proceedings of the International Conference on Intelligent Robots and Systems, pages 595–601, 1994.

[6] W. Burgard, A. Derr, D. Fox, and A.B. Cremers. Integrating global position estimation and position tracking for mobile robots: The dynamic markov localization approach. In Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'98), 1998.

[7] S. Engelson and D. McDermott. Error correction in mobile robot map learning. In Proceedings of the 1992 IEEE International Conference on Robotics and Automation, pages 2555–2560, Nice, France, May 1992.

[8] S. Kwon, K.W. Yang and S. Park. An Effective Kalman Filter Localization Method for Mobile Robots. Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on, pages 1524-1529, Oct. 2006.

[9] T.H. Cong, Y.J. Kim and M. Lim. Hybrid Extended Kalman Filter-based localization with a highly accurate odometry model of a mobile robot. Control, Automation and Systems, 2008. ICCAS 2008. International Conference on 14-17 Oct. 2008. Page(s):738 – 743.

[10] F. Kong, Y. Chen, J. Xie, G. Zhang and Zude Zhou. Mobile Robot Localization Based on Extended Kalman Filter. Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on Volume 2, Page(s):9242 – 9246.

[11] C. Takenga, T. Peng and K. Kyamakya. Post-processing of Fingerprint Localization using Kalman Filter and Map-matching Techniques. Advanced

Communication Technology, The 9th International Conference on Volume 3, 12-14 Feb. 2007, Page(s):2029 – 2034.

[12] A.C. Schultz and W. Adams. Continuous localization using evidence grids. Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on Volume 4, 16-20 May 1998, Page(s):2833 - 2839 vol.4.

[13] A Howard, M.J. Mataric and G.S. Sukhatme. Cooperative relative localization for mobile robot teams: An ego-centric approach. In Proc. of The naval Reasearch Laboratory Workshop on Multi-Robot Systems, Washington, D.C. 2003.

[14] W. Burgard, A.B. Cremers, D. Fox, D. Hahnel, G. Lakemeyer, D. Schulz, W. Steiner and S. Thrun. Experiences with an interactive museum tour-guide robot. Artificial Intelligence 114:3-55, 1999.

[15] T. Rofer and M. Jungel. Vision-Based Fast and Reactive Monte-Carlo Localization. Proc. of the 2003 IEEE International Conference on Robotics & Automation, Taipei, Taiwan, September, 2003.

[16] J. Liu, K. Yuan, W. Zou, and Q. Yang. Monte Carlo Multi-Robot Localization Based on Grid Cells and Characteristic particles. Proceedings of the 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Monterey, California, USA, 24-28, July, 2005.

[17] X. Ma, X. Dai and W. Shang. Vision-based Extended Monte Carlo Localization for Mobile Robot. Proc. of the IEEE International Conference on Mechatronics & Automation, Niagara Falls, Canada, 2005.

[18] A Gasparri, S. Panzieri, F. Pascucci and G. Ulivi, A Hybrid Active Global Localization Algorithm for Mobile Robots. International Conference on Robotics and Automation, Roma, Italy, 10-14 April 2007.

[19] X. Zhang, X. Chen, J. Li and X. Li. Vision-based Monte Carlo – Kalman Localization in a Known Dynamic Environment. Control. Automation, Robotics and Vision, 2006. ICARCV 06. 9 th International Conference on Volume, Issue, 5-8 Dec. 2006.

[20] D.B. Rubin. Using the SIR algorithm to simulate posterior distributions. In M.H. Bernardo, K.M. an DeGroot, D.V. Lindley, and A.F.M. Smith, editors, Bayesian Statistics 3. Oxford University Press, Oxford, UK, 1988.

[21] S. Lenser, and M.Veloso, "Sensor Resetting Localization for Poorly Modeled Mobile Robots"Proceedings of ICRA 2000, IEEE, 2000.

[22] J.S. Gutmann, and D. Fox, "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 October 2002.

[23] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, Robust monte carlo localization for mobile robots. Artificial Intelligence 128,1-2 (2001), 99–141. Available at: http://citeseer.nj.nec.com/thrun01robust.html

[24] A. Singhal, Issues in Autonomous Mobile Robot Navigation (1997). Available at: http://citeseer.nj.nec.com/singhal97issue.html

[25] I. J. Cox and G. Wilfong (Editors), Autonomous Robot Vehicles, Springer-Verlag, New York (1990).

[26] S. Roumeliotis, Reliable Mobile Robot Localization (1999). Available at: http://www-users.cs.umn.edu/~stergios/

[27] J. Borenstein and L. Feng, Measurement and correction of systematic odometry errors in mobile robots. IEEE Transactions on Robotics and Automation 12 (1996), 869–880. Available at: http://citeseer.nj.nec.com/borenstein96measurement.html

[28] J. Borenstein, Control and kinematic design of multi degree of freedom robots with compliant linkage. IEEE Transactions on Roboticsand Automation (1995). Available at: http://citeseer.nj.nec.com/borenstein95control.html

[29] S. Shoval and J. Borenstein, Measurement Of Angular Position Of A Mobile Robot Using Ultrasonic Sensors (1999). Available at: http://citeseer.nj.nec.com/shoval99measurement.html

[30] P. H. Dana, The Global Positioning System (2000). Available at http://www.colorado.edu/geography/gcraft/notes/gps/gps_f.html

[31] S. Thrun, Bayesian landmark learning for mobile robot localization. Machine Learning 33, 1 (1998), 41–76.

[32] J. Manyika and H. Durrant-Whyte, Data Fusion and Sensor Management, a decentralized information-theoretic approach, Ellis Horwood Limited, Chichester, West Sussex (1994).

[33] E. Weisstein, MathWorld (2003). Available at http://mathworld.wolfram.com/.

[34] H. Bruyninkx, Bayesian probability (2002). Available at http://people.mech.kuleuven.ac.be/~bruyninc.

[35] S. Thrun, D. Fox, and W. Burgard, Probabilistic methods for state estimation in robotics. Proceedings of the Workshop SOAVE'97. VDI-Verlag (1997), page: 195–202.

[36] S. Thrun, Probabilistic algorithms in robotics. AI Magazine 21,4 (2000), page: 93–109.

[37] H. Baltzakis and P. Trahanias, Hybrid mobile robot localization using switching state-space models. Proceedings of the 2002 IEEE International Conference on Robotics and Automation. Washington D.C.,USA (2002), page: 366–373.

[38] A. R. Cassandra, L. P. Kaelbling, and J. A. Kurien, Acting under uncertainty: Discrete bayesian models for mobile robot navigation. Proceedings of IEEE/RSJ International Conference on intelligent Robots and Systems (1996).

[39] W. Burgard, D. Fox, D. Hennig, and T. Schmidt, Estimating the absolute position of a mobile robot using position probability grids. Proceedings of the Fourtheenth National Conference on Artificial Intelligence (1996), Page: 896–901.

[40] D. Fox, S. Thrun, W. Burgard, and F. Dellaert, Particle filters for mobile robot localization. Sequential Monte Carlo Methods in Practice. Springer. New York (2001). Available at: http://citeseer.nj.nec.com/fox01particle.html

[41] M. Grewal and A. Andrews, Kalman filtering: theory and practice, Prentice-Hall, Inc., Englewood Cliffs, New Jersey (1993).

[42] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In Proceedings of the IEEE International Conference on Robotics andAutomation (ICRA), 1999.

[43] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In Proceedings of the National Conference on Artificial Intelligence (AAAI), Orlando, FL, 1999. AAAI.

[44] T. L. Dean and M. Boddy. An analysis of time-dependent planning. In Proceeding of Seventh National Conference on Artificial Intelligence AAAI-92, pages 49–54, Menlo Park, CA, 1988. AAAI, AAAI Press/The MIT Press.

[45] S. Zilberstein and S. Russell. Approximate reasoning using anytime algorithms. In S. Natarajan, editor, Imprecise and Approximate Computation. Kluwer Academic Publishers, Dordrecht, 1995.

[46] S. Thrun. Probabilistic robotics. Communications of the ACM, 45(3):52-57, 2002.

[47] S. Thrun, W. Burgard, and D. Fox. 2005. Probabilistic Robotics. MIT Press.

[48] T. Graepel. Statistical physics of clustering algortihms. Technical Report 171822, FB Physik, Institut fur Theoretische Physic, 1998.

[49] S. Theodoridis and K. Koutroumbas, Pattern Recognition. Academic Press, 2006.

[50] Tun Yang, and Victor Aitken, "Uniform Clustered Particle Filtering for Robot Localization" 2005 American Control Conference, June 8-10, 2005. Portland, OR, USA

VITA AUCTORIS

NAME	Ali Akhavan Malayeri
PLACE OF BIRTH	Tehran, Iran
YEAR OF BIRTH	1976
EDUCATION	School of Software Engineering
	Azad University
	Central Tehran branch, Iran
	1996 – 2001 B.Eng.
	School of Computer Science
	University of Windsor
	Windsor, Ontario, Canada
	2007 – 2010 M. Sc.