2011

An Improved Clustering based Monte Carlo Localization approach for Cooperative Multi-robot Localization

Guanghui Luo
University of Windsor

An Improved Clustering based Monte Carlo Localization Approach for Cooperative Multi-robot Localization

by

Guanghui Luo

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

2011

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

This thesis describes an approach for cooperative multi-robot localization based on probabilistic method (Monte Carlo Localization) used in assistant robots which are capable of sensing and communicating one with another. In our approach, each of the robots maintains its own clustering based MCL algorithm, and communicates with each other whenever it detects another robot. We develop a new information exchange mechanism, which makes use of the information extracted from the clustering component, to synchronize the beliefs of detected robots. By avoiding unnecessary information exchange whenever detection occurs through a belief comparison, our approach can solve the delayed integration problem to improve the effectiveness and efficiency of multi-robot localization. This approach has been tested in both real and simulated environments. Compared with single robot localization, the experimental results demonstrate that our approach can notably improve the performance, especially when the environments are highly symmetric.

Keywords: multi-robot, localization, Monte Carlo, belief, clustering

DEDICATION

This thesis is dedicated to my families who have supported me all the way since the beginning of my studies with patience, understanding, and love.

Also, this thesis is dedicated to those lovely people who gave me so many unforgettable memories.

ACKNOWLEDGEMENTS

First and foremost, I would like to heartily thank my supervisor, Dr. Dan Wu, a respectable, responsible and resourceful scholar, who has provided me with valuable guidance in every stage of the writing of this thesis. Without his endless help, this thesis would not have been possible.

I also would like to thank my internal reader, Dr. Imran Ahmad, my external reader, Dr. Huapeng Wu, and my thesis committee chair, Dr.Yung Tsin for spending their time in reviewing this thesis and giving me all the valuable comments.

As well as, a special thanks to Dr. Angela Sodan, who gave me professionally knowledge during the past two years of master's study. I have benefited so much from her courses and constant encouragement. The sudden passing of her really saddens me. I will always miss her. My sincere sympathies go to her family and friends.

Finally, I would like to extend my sincere thanks to Chong Fu, Yuefeng Wang, and many other friends helping me in solving many difficult problems.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Mobile robot localization, the process of determining the position and orientation (pose) of a robot within its operating environment from sensor readings, is a prerequisite for subsequent high level navigation tasks. It has been seen as one of the fundamental problems in mobile robotics [11]. The most widely studied localization problems are: Local localization (position tracking), the most simple localization problem, which is to compensate incremental errors in a robot's motion under the assumption that the initial position is known as prior, and the more challenge one is global localization, in which robots are required to estimate their pose by local and incomplete observed information under the condition of uncertain initial position [14]. Robots need to handle multiple, distinct hypothesis due to the global uncertainty.

During the past two decades, much work has focused on single robot global localization and many probabilistic approaches have been applied with remarkable success, such as Grid-based approaches [1,21], Monte Carlo Localization (MCL) [9,11], and multi-hypothesis approaches [3,22]. The key idea of these probabilistic approaches is to represent the uncertainty of a robot's pose by using probability distributions over the whole space of robot's possible poses instead of relying on a single best guess [19]. Among these probabilistic approaches, Monte Carlo Localization, approximated Bayes Filters using random samples for posterior estimation, achieves a great success for global localization in a highly robust and efficient way.

However, in the latest years, more and more researchers are interested in using multiple mobile robots to improve efficiency and robustness in performing tasks.

Knowing their global positions is also of great importance in multi-robot system. Compared to single robot system, multi-robot system has some obvious advantages. For example, sharing sensor information collected from different robots will increase the robustness of the localization algorithm for each robot. Another advantage is the capability of exchanging their pose information with each other can gain more reference information for localization from their geometric relationship. Moreover, the possibility of heterogeneous group of robots equipped with different sensing devices can achieve more comprehensive environment description [11].

A relatively simple approach for multi-robot localization is to estimate their positions independently. Each robot relies on its own resources rather than combining the experiences from different entities of the team. Nonetheless, an alternative approach, named Cooperative Localization whereby the members of a team of robots estimate one another's position, can obtain better localization performance. So a lot of estimation algorithms focused on fusion of information from multiple robots have been proposed such as Extended Kalman Filter (EKF) [16,17], Maximum Likelihood Estimation (MLE) [2], and Particle Filters [7]. Most of EKF-based and MLE-based approaches require a fusion center to process all the information communication, which makes them susceptible to single-point failures. In addition, applications of these centralized approaches are limited to small group of robots due to high cost of computation and communication. The Particle Filters approach proposed in [7] is based on Markov localization, which can approximate a wide range of belief functions in real-time implementation because of its probabilistic nature. The "detection models" enable robots with the ability of recognizing each other. These detection models are used to synchronize the individual robots' beliefs whenever detection occurs so that to reduce the uncertainty of both robots during localization. However, this approach suffers from the problem of delayed integration, which means the instant updating beliefs of both detected robots may

not contribute positively to the localization process if both beliefs are with highly uncertainty.

In this thesis, we propose an efficient probabilistic approach for cooperative multi-robot localization in indoor environments. Our approach is based on Monte Carlo localization that has been applied with great practical success to single robot localization. The robots, capable of sensing and exchanging information one with another, localize themselves by maintaining their own belief functions which are the clustering based MCL algorithm. Our new developed information exchange mechanism is employed to synchronize each robot's belief whenever one robot detects another in order to speed up the localization process with higher accuracy. Our proposed approach can prevent the localization from suffering the problem of delayed integration by comparing beliefs of both robots at each time of detection to avoid unnecessary information exchange. We utilize the information extracted from clustering component that analysis the distribution of whole particle set to quantify robot's belief and transfer information across different robots. In addition, by analyzing how concentrated the particles are, the robot can carry the notion of whether it has been localized or not by itself instead of incorporating human observers. In our proposed approach, robots themselves are implicitly used as landmarks rather than only external landmarks, therefore, can further facilitate the localization process. Experimental results, carried out in both real and simulated environments using two robots, demonstrate that our proposed approach can significantly reduce the uncertainty compared with single robot localization.

In addition to the introduction chapter, there are four chapters in this thesis. Chapter 2 introduces the materials on which our proposed approach is based, including probabilistic robotics, uncertainty, and mobile robot localization in both single robot and multi-robot system. The details of our proposed approach are discussed in chapter 3. In

chapter 4 we present the information of implementation and experiments in both real and simulated environments. The evaluations of our proposed approach are followed by the experimental results' comparison with single robot localization. Finally, the conclusion and future work are given in chapter 5.

CHAPTER 2

BACKGROUND KNOWLEDGE

This chapter will give the background knowledge that supports the proposed methods. Firstly, we will go through the idea of probabilistic robotics. Then we focus on the problem of single mobile robot localization. The detail of the algorithm named Monte Carlo Localization algorithm will be explained, which is also the core of the proposed approach. Finally, we will review some related works on cooperative multi-robot localization.

2.1 Uncertainty in Robotics

Robotics is the science of perceiving and manipulating the physical world through computer controlled devices; examples of successful robotics systems include mobile platforms for planetary explorations, industrial robotics arms in assembly lines, cars that travel by themselves, and so on [19]. For a robot, it usually consists of the four main components: (1) a physical body, so it can exist in the real world; (2) sensor, so it can sense the environment; (3) effectors and actuators, so it can act; (4) a controller, so it can be autonomous [12].

To do tasks in the real world, robot has to accommodate many uncertainties [19], which are caused by a number of factors. Firstly, the environments of the robot are usually unpredictable especially in the highly dynamic environments such as highways and offices. Secondly the sensors are limited in what they can perceive, such as range and resolution limitations. Thirdly the motor used for the robot actuation is unpredictable.

Control noise and mechanical failure always cause uncertainty. Fourthly, the software of the robot may also create uncertainty, since the internal models of the physical world are approximate and partial. Finally algorithmic approximations are other factor that arise the uncertainty. In real-time systems, accuracy sometimes has to be sacrificed in order to achieve timely response.

As robotics systems have been widely used in the world around us and became more and more important, uncertainty is the major issue for the design of robot systems. How to manage the uncertainty is the first step towards the robust robot systems.

2.2 Probabilistic Robotics

Probabilistic robotics is a relatively new approach to address the uncertainty problem existing in robot's perception and action. The main concept of probabilistic robotics is to represent uncertainty using probability theory.

2.2.1 State

In probabilistic robotics, the *environment* is a dynamical system that consists of internal state. Robot can perceive the information about its environment through sensors, and the robot can also affect the environment through its actuators. Since the enormous of uncertainty exists, robot needs to maintain an internal belief about the state of its environment. The *state* is defined as the collection of all aspects of the robot and its environment that can impact the future. It has been categorized into two groups. State variable that change along with time is called *dynamic state*, such as walking people around the robot. In contrast those states that tend to stay static or no changing at all are

called *static state*, such as the location of walls in buildings. The variables of robot itself, such as its pose, velocity, are also involved in state.

Typically variables describe the state of a robot includes: (1) robot's pose, which is specified by a two-dimensional coordinate and heading direction; (2) in robot manipulation, the pose includes variables for the configuration of the robot's actuators, which are usually referred to as kinematics state; (3) robot's velocity and the velocities of its joints are often referred to as dynamic state; (4) the location and features of surrounding objects in the environment are also considered as state variables. An object may be a desk or a box, and features may be the visual perception such as color or texture; (5) locations and velocities of moving objects and people may be state variables too. Since the list of potential state variables is endless, robot environments can be described using hundreds of and thousands of state variables depending on the requirement of the granularity of environment description.

A state is called *complete* if the current state is the best predicator of the future, which means there is no other knowledge of past states, measurements, or controls can help to refine the future prediction. But in practice it is impossible to find a complete state for any robot systems. A complete state not only contains all aspects of the environment that may influence the future, but also the variables of robot itself. Sometimes there is no chance to reach all these information.

In this thesis, state is denoted x; the state at time t is denoted x_t, and time is discrete. So that all states over time can be described at discrete time steps $t = 0, 1, 2, 3 \cdots$.

2.2.2 Environment Interaction

As the environment is a dynamic system represented by state, which includes the robot itself, the robot can affect the state in two ways: Environment sensor measurements and Control actions. In the first type of interaction, the robot can obtain the state information through its sensors, such as taking a camera image, laser range scanning. The result of these kinds of perception is called a measurement. One feature of this kind of interaction is that the information obtained by sensor measurement usually represent the state moments ago due to some delay. In the second type of interaction, the robot can affect the state of its environment through its actuators. For example, the robot moves to a new location and manipulates objects in the environment. Such action will change the state of the environment. Since the environment usually changes, we assume the robot always performs a control action, even if the robot does not move any of its actuators. In practice, these two types of environment interaction are performed continuously and concurrently by the robot.

According to the two types of interaction the robot hypothetically carries a record of all past sensor measurements and control actions, which is referred to as the measurement data and control data. Measurement data describe a momentary state of the environment. Normally it is under the assumption that the robot takes exactly one measurement at one time. So the measurement data at time *t* will be denoted as z_t. And the notation $z_{t1}: z_{t2} = z_{t1}, z_{t1+1}, z_{t1+2}, \cdots, z_{t2}$ denotes the measurement data obtained from time t_1 to time t_2. Control data provides information about the change of state in the environment. One typical example is the velocity of a robot. The alternative source of control data is odometer. Even though they are sensors that measure the revolution of robot's wheels, they are treated as control data because the measure of which describe the effect of a control action. Control data will be denoted as u_t. In the same way with measurement

data, the notation $u_{t1}{:}\,u_{t2} = u_{t1},\ u_{t1+1},\ u_{t1+2},\ \cdots,\ u_{t2}$ denote the control data from time t_1 to time t_2.

Both measurement and control are very important in a robotic system, but they are with different functions. The measurement tends to increase the robot's knowledge through carrying the state information about its environment. On the contrary, control action tends to produce a loss of knowledge because of the notion of uncertainty that is caused by the motors and the stochasticity of robot environment.

2.2.3 Probabilistic Generative Laws

The evolution of state is given by the probabilistic laws [19]. According to the definition of state, the state x_t is conditioned on all past states, measurements, and controls, it can be described by a probability distribution from this form: $p(x_t|x_{0:t-1}, z_{1:t-1}, u_{1:t})$. And here we assume the sequence of environment interactions is that a measurement z_t is followed by a control action u_t. Notice that a complete state means it is the best predicator for the future. In particular, x_{t-1} is a sufficient summary of all previous measurements and controls up to time $t-1$. And only the current control data u_t can influence the previous expression if we are given the complete state x_{t-1}. So the expression has the following equality: $p(x_t|x_{0:t-1}, z_{1:t-1}, u_{1:t}) = p(x_t|x_{t-1}, u_t)$. The property used in this expression is called conditional independence, which means if we know the conditioning variables, such as the x_{t-1}, u_t, then the certain variables, here is x_t, are independent of other variables $z_{1:t-1}$ and $u_{1:t-1}$. Likewise, the conditional independence also works for the process of modelling the measurements if x_t is complete: $p(z_t|x_{0:t}, z_{1:t-1}, u_{1:t}) = p(z_t|x_t)$. Here the complete state x_t is sufficient to predict the measurement z_t, which means all other variables, such as all past measurements, controls, and past states, are irrelevant.

The probability $p(x_t|x_{t-1}, u_t)$ is called *state transition probability* or *motion model*. It shows how state x_t evolves over time conditioned on the controls u_t. The probability $p(z_t|x_t)$ is called *measurement probability* or *measurement model*, which specifies how measurements z is generated from state x. The dynamical system of the robot and its environment is presented by the combination of motion model and measurement model. Figure 2.1 illustrates the evolution of states and measurements. The state x_t stochastically depends on the previous state x_{t-1} and the control u_t. The measurement z_t stochastically depends on the state x_t. This model shown in Figure 2.1 is also known as hidden Markov model or dynamic Bayes network.

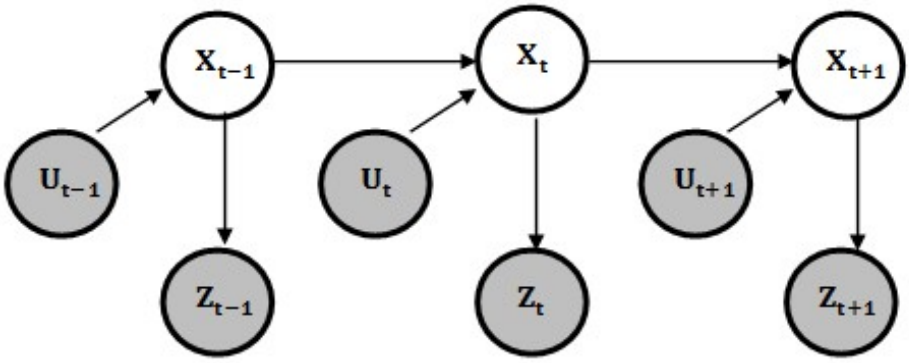


Figure 2.1: The dynamic Bayes network that characterizes the evolution of control, states and measurements [19].

2.2.4 Belief

Another important concept in probabilistic robotics is *belief*, which is used to reflect the robot's internal knowledge about the state of the environment [19]. However the state usually cannot be measured directly, the robot must infer its pose from the data collected. The way to represent belief in probabilistic robotics is through conditional probability

distributions. The density value is assigned to each possible state hypothesis with regards to the true state by belief distribution. The expression of belief over state x_t at time t is $bel(x_t) = p(x_t|z_{1:t}, u_{1:t})$. It is a posterior probability over states conditioned on all the past control data and all the past measurement data. This expression represents the posterior probability including incorporating the most recent measurement data z_t, but occasionally a probability distribution without incorporating the measurement data z_t called prediction is proven to be useful, which is denoted as $\overline{bel}(x_t) = p(x_t|z_{1:t-1}, u_{1:t})$. It reflects the prediction of the state x_t on the previous state x_{t-1} before incorporating the measurement at time t. In the next we need to calculate the $bel(x_t)$ from $\overline{bel}(x_t)$ by incorporating z_t, which is called measurement update.

2.2.5 Bayes Filter

Bayes Filter algorithm is the most general algorithm used for belief calculating. The fundamental components in Bayes Filter algorithm are the two conditional probabilities we mentioned above: motion model and measurement model. This algorithm calculates the belief distribution based on control data and measurement data. Table 2.1 is the pseudo code of the basic Bayes Filter algorithm. From the pseudo code, we can find that Bayes filter is a recursive algorithm. In one iteration the belief $bel(x_t)$ is calculated from the previous belief $bel(x_{t-1})$. The inputs are the belief at time $t-1$, the most recent control data u_t, and the most recent measurement data z_t. It outputs the belief $bel(x_t)$ at time t. The calculating process can be divided into two steps: prediction (line 3) and measurement update (line 4).

In prediction phase, it processes the control model, which calculates the prediction belief $\overline{bel}(x_t)$ before incorporating the measurement at time t. It is obtained by the

integral of the product of two probability distributions: the prior assigned to the previous state posterior x_{t-1}, and the probability that control data u_t may induce a state transition from the all the hypothesis states at time $t-1$ to time t. In the measurement update phase, it multiply the prediction belief $\overline{bel}(x_t)$ by the probability of measurement model $p(z_t|x_t)$ that the measurement data z_t may be observed at state x_t. The constant η is used to normalize the resulting product, since it may not integrate to 1. Finally line 6 returns the normalized belief $bel(x_t)$.

1:	**Algorithm Bayes_Filter** $(\mathrm{bel}(x_{t-1}),\ u_t,\ z_t)$:
2:	for all x_t do
3:	$\overline{\mathrm{bel}}(x_t) = \int p(x_t \mid u_t, x_{t-1})\ \mathrm{bel}(x_{t-1}) dx_{t-1}$
4:	$\mathrm{bel}(x_t) = \eta\ p(z_t \mid x_t)\ \overline{\mathrm{bel}}(x_t)$
5:	endfor
6:	return $\mathrm{bel}(x_t)$

Table 2.1: The Bayes Filter algorithm [19].

As the Bayes filter algorithm calculates the belief recursively, it demands an initial belief $bel(x_0)$ at time $t = 0$. If the initial state x_0 is given, then $bel(x_0)$ is initialized with all probability concentrated on a point of the given value of x_0, and zeroizes anywhere else. If the initial state x_0 is unknown, then we use a uniform distribution over all possible value of x_0 to represent $bel(x_0)$. If the initial state x_0 is partially known, a non-uniform distribution could be used.

In probabilistic robotics, many different implementations of Bayes filter algorithm vary in different assumptions of the measurement probability, the state transition probability, and the distribution of belief. Throughout the whole thesis, we calculate the belief distribution under the Markov assumption, which postulates that past and future data are independent if the current state x_t is known. It means no past measurement or control data would provide us additional information to the stochastic evolution of future

states if we knew the current belief $bel(x_t)$.

2.3 Mobile Robot Localization

Mobile robot localization is the problem of determining the pose of a robot relative to a given map of the environment [19]. It is an instance of general localization problem, and Bayes filter algorithm is widely used in this position estimate. Mobile robot localization is the most important problem in robotics system, since all tasks and navigations are started from knowing the pose of the robot or the location of objects that need to be manipulated. Normally a robot cannot perceive its pose directly relative to the global coordinate system. Instead, its pose information can be inferred from the measurement data and control data given the map of the environment. A graphical model of mobile robot localization problem is given by Figure 2.2. The shaded nodes represent given information including the measurement data z, the control data u, and the given map of the environment. The goal is to estimate its position, which is represented by the white nodes.

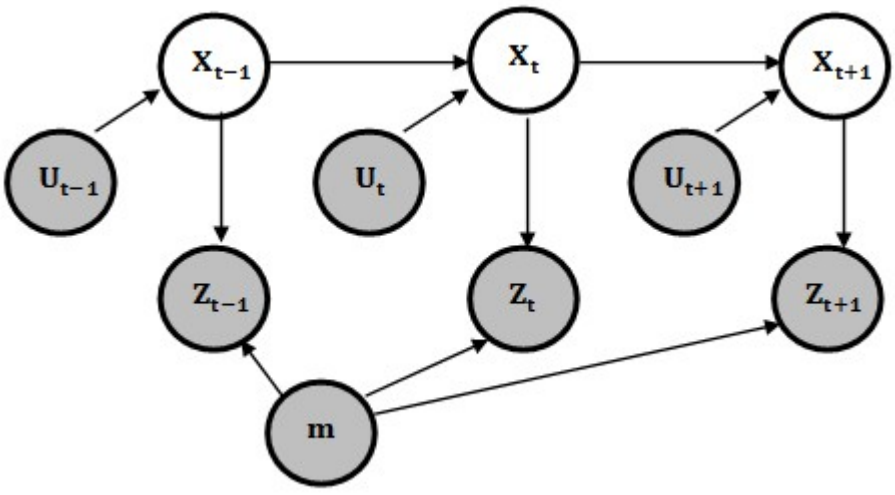


Figure 2.2: Graphical model of mobile robot localization [19].

2.3.1 Categories of Localization Problems

The mobile robot localization problem can be divided into many types according to the nature of the environment and the initial knowledge that a robot may have relative to the given map. Here we give a brief introduction of the taxonomy of localization problems.

The first class of the localization problems is categorized by the different types of initial knowledge of a robot. Under this class, two kinds of localization problems have been widely studied: local localization (position tracking) and global localization. The former one is the simplest one since the initial pose of the robot is given and then the problem is to compensate the incremental noise in robot's odometry. Notice that the noise is usually assumed to be small, and the uncertainty is local and confined to the area where is closed to the true location of the robot. In this case, the uncertainty can be modelled by a unimodal distribution like Gaussian. The latter one is more difficult since the robot is not been told its initial pose, instead it has to determine it from scratch. One variant of the global localization is called kidnapped robot problem where the robot might be kidnapped during the operation and taken to somewhere in its environment without notified. It is even more difficult because even the most state-of-the-art localization algorithms will fail to locate sometimes. Therefore the ability to recover from failures is the sign of truly autonomous robots.

The second class is categorized by the property of environment. Environment can be static or dynamic. Static environment means only the robot can move in this environment while all other objects remain stationary. On the contrary, in dynamic environment not only the robot itself, the locations of other objects or configurations will change over time. Examples of the changes are people, movable furniture, or animals. Localization in

dynamic environment is more difficult due to the burden of dealing with those changing states in the environment.

The third class is characterized by the localization algorithm pertaining to whether or not the algorithm itself controls the motion of the robot: passive localization and active localization. In passive localization, the algorithm only observes the robot operating, and does not control the motion to facilitate the localization. Usually the robot will move randomly. In active localization, the algorithm always controls the robot motion to achieve the goal of minimizing the localization errors. Obviously, active localization approaches outperform passive ones in yielding better results.

The last class of the localization problem is related to the number of robots involved: single-robot localization and multi-robot localization. The former one is most studied in recent years. There is no communication problem in this case due to only one robot is considered and all data is integrated only in one robot platform. In multi-robot localization, more than one robot is involved. The problem comes in how to represent the multiple belief system and the communication between them.

In this thesis, we focus on the global passive localization for multi-robot in the static environment.

2.3.2 Localization Algorithms

As we mentioned before the processing of localization is the first step of almost all robotics systems, many probabilistic approaches have been developed to address the various problems in localization, especially for single robot localization. The most commonly used algorithms are Kalman filter-based localization, Multi-hypothesis

tracking filter, Grid localization and Monte Carlo localization. The Kalman filter-based localization algorithms are based on the assumption that the uncertainty in the robot's pose can be represented by a unimodal Gaussian distribution. In addition they take advantage of a range of restrictive assumptions for example Gaussian distribution noise and Gaussian distributed initial uncertainty. Therefore Kalman filter-based approaches work exceedingly well for position tracking problem. On the other hand the Kalman filter-based approaches are not suitable for global localization which needs to represent the uncertainty in multi-modal distribution. To overcome this limitation, Multi-hypothesis tracking filters have been introduced to solve the global localization by representing the uncertainty using multiple Gaussians, which is mixture of normal distributions, but with a high burden of computation. Grid localization approaches use a histogram filter to represent the posterior belief over a grid decomposition of the pose space [21]. The posterior belief for each grid is given by piecewise constant functions. These grid based algorithms are capable of solving the global localization problems and non Gaussian probability distribution. Nevertheless, uncertainty representing in this way will face a trade-off between accuracy and computational efficiency. The more number of evenly spaced grids means the smaller approximation error, but require higher computational complexity. Monte Carlo localization is an alternative approach for global localization problem with reasonable cost of computational complexity. Our proposed approach is based on the Monte Carlo localization, and we will give details in the following section.

2.3.3 Monte Carlo Localization

Monte Carlo Localization (MCL) is one of the latest and commonly used probabilistic approaches to single robot localization. MCL is an implementation of Bayes Filter. Instead of describing the probability density function itself, MCL represents uncertainty by maintaining a set of samples that are randomly drawn from the probability

density [19]. It is applicable to both local and global localization problems.

The filter representing posteriors by finitely many samples is known as particle filter [19]. Each particle drawn according to the posterior distribution represents a possible pose of the robot. Table 2.2 shows the particle filter algorithm. It possesses three steps: sampling, importance weighting and resampling. The inputs are the particle set X_{t-1}, together with the most recent control data u_t and the most recent measurement data z_t. In the first step (line 3), a hypothetical sample set $X_t^{[n]}$ is generated based on the state transition probability $\mathrm{p}(u_t, X_{t-1}^{[n]})$ for time t. The importance weighting step shows in line 4, the importance factors $w_t^{[n]}$ for each particle are calculated using the probability model of sensor measurement $\mathrm{p}(z_t | X_t^{[n]})$. These importance factors are used to incorporate the measurement z_t into the particle set. Then in the last step (from line 7 to line 10), it draws with replacement N particles from the temporary set $X_t^{[n]}$. The probability of drawing each particle is given by its importance weight. The resulting sample set is with the same size to the original sample set. Since the resampling step refocuses the particles to regions in the state space with high posterior probability, it has the effect that the more the number of particles falls in a region of the state space the more likely that region represents the true state. Through recursively computing the three steps, it focuses the computational resources of the filter algorithm to regions in the state space where they matter the most [19].

Algorithm Particle Filter (X_{t-1}, u_t, z_t):

1: $\overline{X}_t = X_t = \emptyset$
2: for n =1 to N do
3: sample $X_t^{[n]} \sim p(u_t, X_{t-1}^{[n]})$
4: $w_t^{[n]} = p(z_t | X_t^{[n]})$
5: $\overline{X}_t = \overline{X}_t + < X_t^{[n]}, w_t^{[n]} >$
6: endfor
7: for n = 1 to N do
8: draw i with probability $\propto w_t^{[i]}$
9: add $x_t^{[i]}$ to X_t
10: endfor
11: Return X_t

Table 2.2: The Particle Filter algorithm [19].

The posterior distribution representing in this way has the following advantages: Firstly, particle filters is a nonparametric approach. Therefore it can represent a much broader space of distributions than the previous parametric algorithms, for example, Gaussians. Secondly, particle filters can model nonlinear transformations of random variables, and it can handle multi-modal densities which have been widely used in global localization. Thirdly, compared to grid localization, particle filters only focus the computational resources on the possible poses rather than keeping all the grids status. Finally, the implementation of particle filters is very easy.

Table 2.3 depicts the basic MCL algorithm from the particle filter. The motion model and measurement model here is corresponding to the state transition probabilistic model and sensor measurement probabilistic model in particle filter respectively. The basic MCL algorithm represents the posterior belief $bel(x_t)$ by a sample set of N total particles $X_t = \{x_t^{[1]}, x_t^{[2]}, \cdots, x_t^{[N]}\}$. The inputs of MCL algorithm are the previous particle set X_{t-1}, control data u_t, measurement data z_t, and the given map m of the environment. Line 1 initializes the particle set $\bar{X}_t$ and X_t. And then for each particle, line 3 do sampling from

the motion model, line 4 calculates the importance weight of that particle from the measurement model. From line 7 to line 10, is the resampling phase. The algorithm draws with replacement N particles from the temporary set $\bar{X}_t$. The probability of drawing each particle is given by its importance weight. Finally, return the posterior particle set X_t, which contained the particles that with higher importance weights. Noticed that, if MCL finishes successfully, most particles are concentrated on one small region which represents the location of robot, but there is not such a stop condition in the basic MCL algorithm, which means the recursive algorithm will been executing during the robot's whole life.

Algorithm MCL (X_{t-1}, u_t, z_t, m)

1: $\overline{X}_t = X_t = \emptyset$
2: for n =1 to N do
3: $X_t^{[n]}$ = sample_motion_model (u_t, $X_{t-1}^{[n]}$)
4: $w_t^{[n]}$ = measurement_model (z_t, $X_t^{[n]}$, n)
5: $\overline{X}_t = \overline{X}_t + < X_t^{[n]}, w_t^{[n]} >$
6: endfor
7: for n = 1 to N do
8: draw i with probability $\propto w_t^{[i]}$
9: add $x_t^{[i]}$ to X_t
10: endfor
11: Return X_t

Table 2.3: The Monte Carlo Localization algorithm [19].

As MCL algorithm is the basic component of our proposed approach, each robot perform MCL to model its own belief to the external environment. We will explain the detailed implementations of the three steps of MCL which is motion model, measurement model and the resampling algorithm.

(a) Motion Model

Motion model or the probabilistic kinematic model plays an essential role in the state transition probability $p(x_t|x_{t-1}, u_t)$, which is the sampling phase of MCL. The outcome of robot's motion is uncertain due to control noise or unmodeled effects. Thus, a posterior probability is needed to represent robots' operation.

Robot's motion is the calculus describing the effect of control actions on the configuration of a robot [22]. The configuration is described by a two dimensional coordinates and its angular orientation, which is referred as $pose(x, y, \theta)$ relative to its external planar environment. Then the state transition probability $p(x_t|x_{t-1}, u_t)$ can be explained in this way: Both x_t and x_{t-1} are robot's poses, and u_t is a motion command. The posterior distribution x_t is the outcome of executing a motion u_t at x_{t-1}. In this thesis, we choose odometry model to represent the probabilistic motion model, which is u_t. The odometry motion model is usually obtained by integrating wheel encoder information. It is specified by a rotation, followed by a translation and then a second rotation in time interval $(t-1, t]$ shown in Figure 2.3.

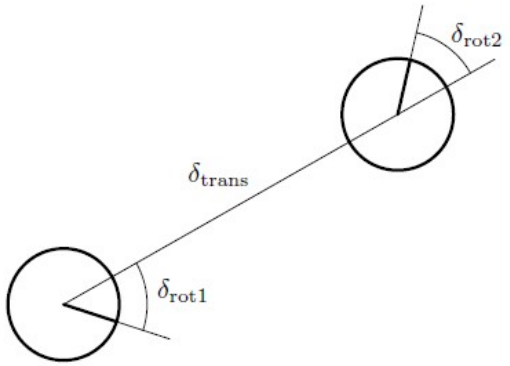


Figure 2.3: The odometry motion model [22].

Notice that any drift or slippage will cause error to the posterior distribution. Therefore these noises should be appropriately modeled. A sampling approach that calculates probability $p(x_t|x_{t-1}, u_t)$, named *sample_motion_model_odometry*, is shown in Table 2.4. The two inputs are initial pose x_{t-1} and an odometry data u_t, and output a randomly guessed poses x_t distributed according to $p(x_t|x_{t-1}, u_t)$. Lines 1 to 3 recover the odometry data from sensor readings. Lines 4 to 6 calculate the motion with noises. Four error parameters are used to control the noises. The parameters α_1 and α_4 specify the noise in rotation. The parameters α_2 and α_3 specify the noise in robots' translation. Lines 7 to 10 return a random poses x_t.

Algorithm sample_motion_model_odometry (u_t, x_{t-1})***:***

1: $\delta_{\text{rot1}} = \text{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta}$
2: $\delta_{\text{trans}} = \sqrt{(\bar{x} - \bar{x}')^2 + (\bar{y} - \bar{y}')^2}$
3: $\delta_{\text{rot2}} = \bar{\theta}' - \bar{\theta} - \delta_{\text{rot1}}$

4: $\hat{\delta}_{\text{rot1}} = \delta_{\text{rot1}} - \text{sample}(\alpha_1 \delta^2_{\text{rot1}} + \alpha_2 \delta^2_{\text{trans}})$
5: $\hat{\delta}_{\text{trans}} = \delta_{\text{trans}} - \text{sample}(\alpha_3 \delta^2_{\text{trans}} + \alpha_4 \delta^2_{\text{rot1}} + \alpha_4 \delta^2_{\text{rot2}})$
6: $\hat{\delta}_{\text{rot2}} = \delta_{\text{rot2}} - \text{sample}(\alpha_1 \delta^2_{\text{rot2}} + \alpha_2 \delta^2_{\text{trans}})$

7: $x' = x + \hat{\delta}_{\text{trans}} \cos(\theta + \hat{\delta}_{\text{rot1}})$
8: $y' = y + \hat{\delta}_{\text{trans}} \sin(\theta + \hat{\delta}_{\text{rot1}})$
9: $\theta' = \theta + \hat{\delta}_{\text{rot1}} + \hat{\delta}_{\text{rot2}}$

10: return $x_t = (x', y', \theta')^T$

Table 2.4: The sample odometry motion model algorithm [22].

Figure 2.4 shows examples for different values of the error parameters from the odometry motion model. Figure 2.4(a) is a typical one. Figure 2.4(b) and (c) show unusually large value of transition error and rotation error, respectively. Figure 2.5 is an example of the odometry motion model "in action" by superimposing sample sets from

multiple time steps. The robots' motion follows the solid line, the particles distributed at different time steps show the robots' belief. The uncertainty grows quickly as the robot does not sense its external environment.

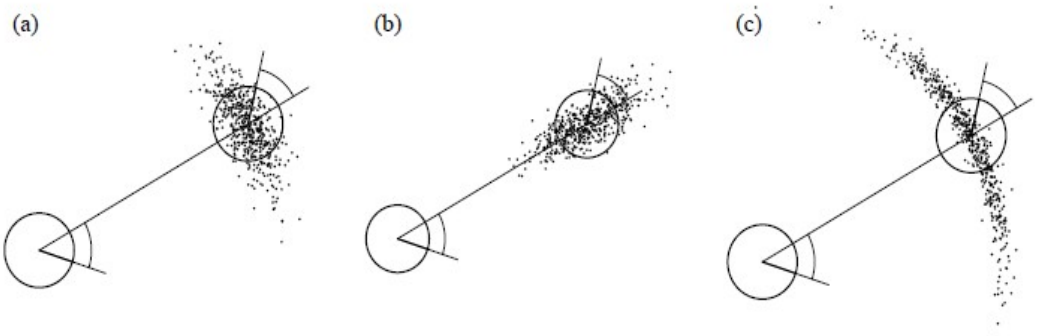


Figure 2.4: Sampling from the odometry motion model, each diagram shows 500 samples [22].

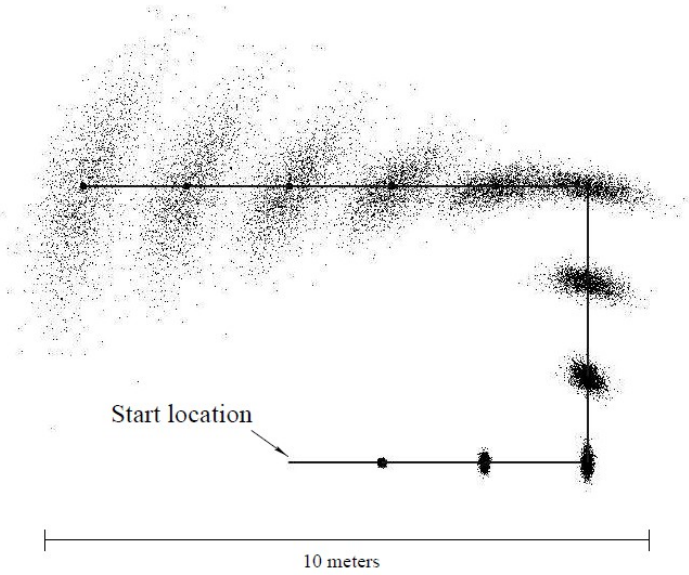


Figure 2.5: Sampling approximation of the position belief for a non-sensing robot [22].

(b) Measurement Model

The second phase of MCL is importance weighting, which is also call measurement

update. The corresponding model is called measurement model, denoted as $p(z_t|x_t)$. It describes the formation process by which sensor measurements are generated in the physical world [22]. Many different sensors are used in robots, such as tactile sensors, range sensors, and cameras. The specifics of the model depend on the sensors used in robots. The details of all the sensors equipped on the robot used in our experiments will be given in chapter 4. Here we just discuss the sensors used in measurement model: bumper sensors, wall sensor, and infrared sensor. The first two types of sensors are uses to detect the landmarks such as wall or any other obstacles in the environment. If any of these sensors return a positive value then we will assign high probability (high weight) to particles who are around walls and obstacles, and low probability to the rest of particles. The weighted particles are then used in the resampling phase. The infrared sensor is used to detect other robots. We put a virtual wall (a standard Infrared remote transmitter) on the top of each robot. If one robot's infrared sensor (virtual wall sensor) returns a positive value, it will consider it has detected another robot. Then according to our proposed approach, it will come to a decision of whether to allow an information exchange happen.

(c) Resampling algorithm

The last phase of MCL is resampling or importance sampling. The algorithm used for resampling is called *low variance sampling* shown in Table 3.5. Instead of choosing M random numbers and selecting those particles that corresponds to these random numbers, this algorithm computes a single random number and selects samples according to this number but still with a probability proportional to the sample weight [22]. A random number r is drawn from the interval $[0; M^{-1}]$, where M is the number of samples to be drawn at time t. The algorithm selects particles by repeatedly adding fixed amount M^{-1} to r and by choosing the particle that corresponds to the resulting number. The while loop calculates the sum U of resulting number, and checks whether i is the

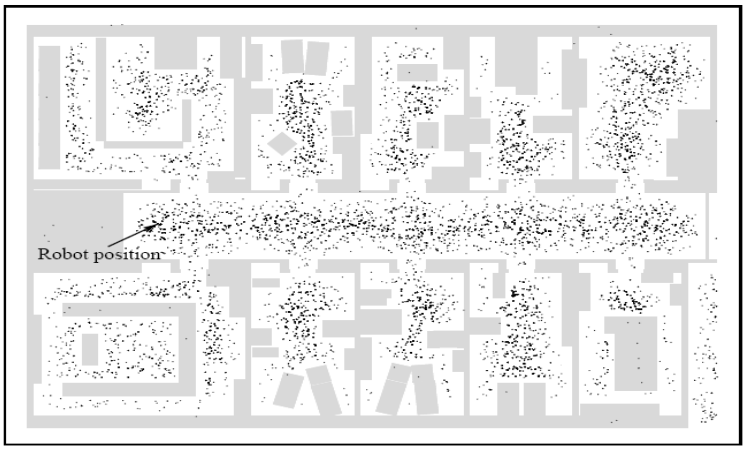
index of the first particle such that the corresponding sum of weights exceeds U. Then the first particle satisfies this condition will be selected. Finally return $\bar{X}_t$ with M particles. It is very efficient since sampling M particles requires $O(M)$ time.

Algorithm Low_variance_sampler (X_t, W_t):

1: $\bar{X}_t = \emptyset$
2: $r = rand(0; M^{-1})$
3: $c = w_t^{[1]}$
4: $i = 1$
5: for $m = 1$ to M do
6: $U = r + (m-1) \cdot M^{-1}$
7: while $U > c$
8: $i = i + 1$
9: $c = c + w_t^{[i]}$
10: endwhile
11: add $x_t^{[i]}$ to $\bar{X}_t$
12: endfor
13: return $\bar{X}_t$

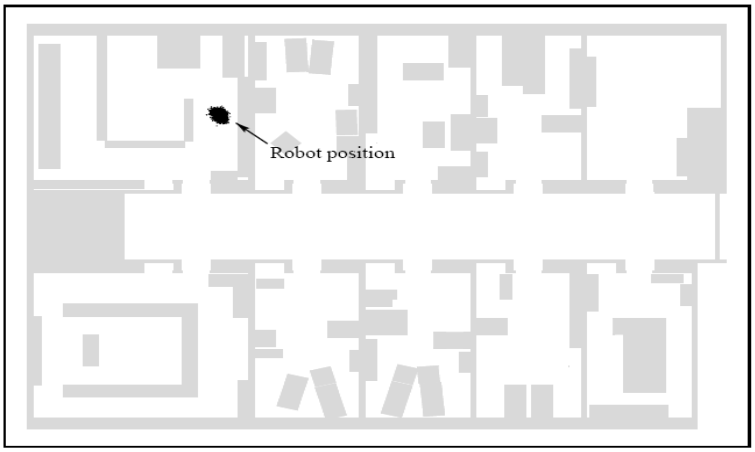
Table 2.5: Low variance resampling for the particle filter [22].

Figure 2.6 shows an example of global localization using MCL in an office environment. The robot is equipped with the sensor of laser range finder. In Figure 2.6(a), at the beginning of global localization all particles are randomly drawn from the uniform distribution over the space. In Figure 2.6(b) shows the distribution of particles after the robot moving approximately 1 meter, we can see all particles are concentrated on two locations due to the symmetry of the environment. In Figure 2.6(c) only one possible location left, since the ambiguity is breaking by entering one specific room.



(a)

(b)



(c)

Figure 2.6: Example of global localization using MCL in an office environment. (a)Global uncertainties in the beginning, all particles are uniformly distributed. (b)After moving approximately 1 meter, most particles are concentrated on two possible locations due to environment symmetry. (c)After entering one room, the ambiguity is resolved; all particles are concentrated on the true pose of the robot [19].

2.3.4 Multi-Robot Localization

Recently, many robotic applications require that robots work cooperatively in order to perform a certain task [17]. Knowing their global positions is the first step in multi-robot system, therefore, a lot of research work start to focus on multi-robot localization. The key idea of multi-robot localization is to integrate measurements taken at different platforms, so that each robot can benefit from data gathered by robots other than itself. There are some existing approaches addressing the problem of integrating the sensor information from multiple robots:

One early cooperative positioning technique with multiple robots proposed by Kurazume and Nagata [15] is known as “portable beacons”. The basic idea underlying this technique is that each robot repeats move-and-stop actions and serves as a landmark for the other robots. A group of robots is divided into two teams. At each time instant, one team is in motion, while the other remains stationary and acts as a landmark. Although this motion scheme can efficiently reduce the positioning error by maintaining knowledge of their positions, the shortcoming is also obvious: it constraints the robots’ motion in the way of allowing only a part of the entire group of robots to move at a certain time instant. Consequently, this approach will dramatically slow down the overall localization speed.

An EKF-based approach to cooperative multi-robot localization was introduced by Roumeliotis [20]. This approach allows all robots in the group to move simultaneously, and to propagate their state and covariance estimates independently by decomposing the centralized EKF-based cooperative localization into N communicating filters. However, during each update cycle, all robots need to communicate with each other and update the covariance matrix for all pose estimates. This induces a computational cost of $O(N^2)$, where N is the number of robots in the team, for processing each relative position

measurement. Considering the total number of robot-to-robot measurements per cycle can be as high as $N(N-1)$, the overall processing cost becomes $O(N^4)$ [8]. The main drawback is that the high cost of computation and communication limits this approach to small robot teams in real-time operation.

Fox and Burgard [7] have proposed a different implementation of cooperative multi-robot localization schema which extends the Monte Carlo Localization algorithm. The sample-based version of Markov localization enables localizing mobile robots in an any-time fashion. This approach avoids the curse of dimensionality (for N robots, one must maintain a $3N$ dimensional distribution) by factoring the distribution into N separate components (one for each robot). Each robot in the system maintains a probability distribution describing its own pose based on odometry data and environment sensing. When one robot detects another, the "detection model" is used to synchronize the individual robots' beliefs, thereby introducing additional probabilistic constrains which ties one robot belief to another robot's belief functions to reduce the uncertainty. In order to synchronize two sample sets drawn randomly by two detected robots, they transform sample sets into density functions using density trees [6] which can integrate information from other robots into a robot's belief more straightforwardly in response to the detection and the belief of the detect robot. One limitation of this approach is that the information exchange between two detected robots cannot ensure it is always benefit the localization process since the belief refining happens immediately whenever detection occurs. For example, if the beliefs of both detected robots are highly uncertain, the current belief refining will not speed up the localization.

CHAPTER 3

A CLUSTERING BASED MCL APPROACH FOR COOPERATIVE MULTI-ROBOT LOCALIZATION

3.1 Motivation

As discussed in chapter 2, many researchers have paid much attention in cooperative multi-robot localization. They all improve the performance either in the aspect of efficiency or accuracy. However, there are some limitations existing in previous approaches. The mover and observer schema in [15] slows down the localization speed due to the restriction of only one robot of a team allowed to move in a time. The proposed method in [20] cannot extend to large group of robots due to high cost of computation and communication. Another approach for collaborative multi-robot localization which is based on MCL [7] suffers from the problem of delayed information exchange, which means robots refine their belief instantly whenever two robots encounter with each other regardless of whether a update will benefit the localization process or not.

As to the basic component of our proposed approach, which is MCL algorithm, many algorithms extending the basic MCL have been presented recently, such as adaptive samples based MCL approach [5], mixture of particle sets based MCL approach [19], reverse MCL approach [11] et cetera. All of these approaches pay attention in improving the accuracy and efficiency of the algorithm. A clustering based MCL algorithm for single mobile robot localization proposed in [6] focuses on the purpose of how to let the robot itself know it has been successfully localized. The motivation of their approach is that human beings can easily know the progress of localization from the distribution of

particles, whereas the robots cannot observe this information directly by the algorithm. Almost all extended MCL approaches inherited this limitation from the basic MCL algorithm. However, the clustering based MCL approach could monitor the stage of localization in real time through employing a clustering component to analyze the distribution of particles so that to help robot to carry the notion of the progress of localization. In other words the information extracted from the clustering component could provide the mathematical information of the uncertainty. Inspired by it, we propose a clustering based MCL for cooperative multi-robot localization through expanding the clustering component to communication between robots. Our approach aims to deal with the problem of delayed information exchange, and still keep the advantage of helping the robot to be aware of the progress of localization.

3.2 Clustering Algorithm

3.2.1 Introduction of Clustering

Clustering is the assignment of a set of observations into subsets (called clusters) so that observations in the same cluster are similar in some sense. And the cluster is defined as “continuous regions of this space containing a relatively high density of points, separated from other high density regions by regions of relatively low density of points” in [4], which is similar to the observations in MCL. After robot moving and perceiving for a certain time, particles will focus on a few different places of the giving map. Those different locations contained many particles could be considered as different clusters. Those particles contained in the same cluster are similar to each other in representing the estimated pose of the robot. In the other hand, those particles within different clusters are dissimilar to each other in pose estimate. In [21] they have defined the clustering in a

mathematical way: X is the data set, contained vectors

$$X = \{x_1, x_2, \cdots x_N\}.$$

Then the *m-clustering* of X, which is the partition of X into m clusters: $C_1, \cdots C_m$ while the following three conditions are satisfied:

- $C_1 \neq \emptyset, i = 1, \cdots, m$
- $\bigcup_{i=1}^{m} C_i = X$
- $C_i \cup C_j = \emptyset, i \neq j, i, j = 1, \cdots, m$

The three conditions means there is at least one cluster. No cluster is empty, and the union of all clusters contain the whole data set, but with no duplicated vectors existed in two clusters. The effect of clustering is that the vectors contained in the same cluster are more similar to each other and less similar to the vectors of other clusters [21].

In order to decide which clusters should be combined, or where a cluster should be split, a measure of dissimilarity between particles or clusters is required, which is called proximity measure or distance measure. The shape of clusters will vary in different proximity measure. As in our experiments all the poses is represented by a two dimensional coordinate system, it's effective for us to choose the Euclidean distance measure $d(x, y) = \sum\sqrt{(x_i - y_i)^2}$ as our proximity measure. Hence the distance between two particles P_i and P_j is calculated by the formula $d(P_i, P_j) = \sum\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. As to the distance between a particle P_i and a cluster C_k contained a certain number of particles, two steps we need. Firstly we need to choose a representative for that cluster C_k, and then the distance $d(P_i, C_k)$ is measured as the proximity between particle P_i and the representative. Figure 3.1 shows three most

commonly used representatives for different shape of clusters: point representative for compact cluster, hyperplane representative for linear cluster, and hyperspherical representative for hyperspherical cluster. In the context of MCL, compact cluster is the one we mostly encountered, thus we choose the point representative to represent a cluster, which is the mean point calculated by $P_{mean} = \frac{1}{N}\sum P_i$, N is the number of particles contained in that cluster.

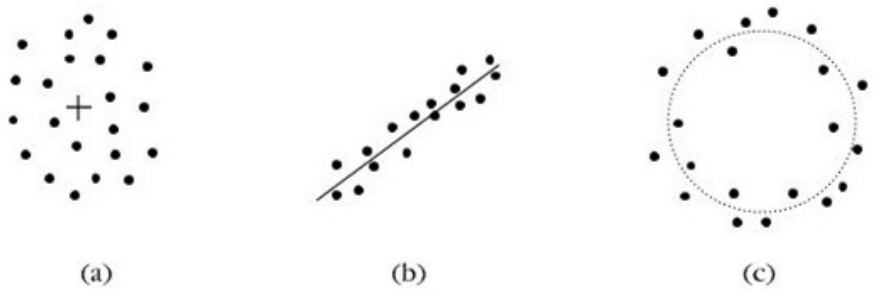


Figure 3.1: Representatives of different shape of clusters. (a) Point representative for compact cluster. (b) Hyperplane representative for linear cluster. (c) Hyperspherical representative for hyperspherical cluster [21].

3.2.2 Details of BSAS algorithm

Various clustering algorithms, such as sequential algorithms, hierarchical clustering algorithms, and clustering algorithms based on cost function optimization, have been applied in dealing with different clustering problems. Notice that different combination of proximity measure and a clustering algorithm will produce different results. As we already known, the real time calculation required in probabilistic robotics drive us to pay attention to the computational complexity. The efficiency is also one of the most important factors for us to choose the clustering algorithm. Therefore we choose one sequential algorithm called *Basic Sequential Algorithm Scheme (BSAS)* [15,21] because it is a fast method and quite straightforward. In addition, it is suitable for producing

compact shape of clusters.

The BSAS has following features: All the vectors (particles in our proposed approach) are presented to the algorithm only once, which means the time complexity for this algorithm is $O(N)$. The number of clusters is not need to be known as prior. New clusters are produced as the algorithm evolves. The sequence of particles that been presented to the algorithm matters the clustering results. Different ordering may lead to totally different number of clusters and the clusters themselves will also differ. Another important feature is that the user defined threshold strongly affects the number of clusters. Either too big or too small will cause unnecessary problems in the experiments. Thus the choice of threshold is highly recommended based on the interpretation the expert or the experiment experience provides. Table 3.1 shows the BSAS written in pseudo code.

Algorithm BSAS ($X(x_i, \dots x_N), \theta$):

1: $m = 1, C_m = \{x_1\}$
2: for *i =2 to N* do
3: find $C_K: d(x_i, C_k) = min_{1 \le j \le m} d(x_i, C_j)$
4: if $d(x_i, C_k) > \theta$ then
5: $m = m + 1,\ C_m = \{x_i\}$
6: else
7: $C_k = C_k \cup \{x_i\}$, update its representative
8: endif
9: endfor

Table 3.1: The Basic Sequential Algorithm Scheme (BSAS) [21].

The BSAS algorithm takes the whole particle set $X(x_i, \dots x_N)$ that need to be clustered and the user defined threshold of dissimilarity θ as input. Line 1 initialize the first cluster, which contains the first particle x_1 presented to the algorithm. From line 2 to line 9 it is a large for loop sequentially going through all the rest particles. The dissimilarity measures between current particle and every existing cluster is calculated in order to find a minimum one in line 3. From line 4 to line 8, if the minimum measure calculated in line 3 is greater than θ, a new cluster that containing current particle will be

created. Otherwise, the considered particle will be assigned to the existing cluster which has the minimum dissimilarity measure to it. And update its representative of this cluster.

3.2.3 Information extracted from clustering

In this section we will discuss the outputs of BSAS algorithm and the corresponding meaning to our proposed approach. Four important variables that can be extracted from BSAS algorithm are: $n_c, pose(a_1, \cdots, a_n)$, n_{max}, and p_{max}.

Firstly, the variable n_c is the number of clusters. It indicates how many clusters exist at current time of clustering. Each cluster contains a certain number of particles. The second variable $pose(a_1, \cdots, a_n)$ is an array which indicates the pose of the representative of each cluster. Each pose can be considered as one possible location of robot. This information is used in process of information exchange explained in 3.3.2. The third variable n_{max} indicates the number of particles contained in the cluster which has the largest number of particles compared to other clusters. The last one p_{max}, calculated from n_{max}, is the percentage of n_{max} in the whole particle set. The percentage number is a more intuitive way that refers to the size of the largest cluster than n_{max} as it is independent to the total number of particles. It also can represent the level of certainty about where it is, since the distribution of particles has the effect that the more number of particles fall in a region, the more likely it represent the true pose of robot.

The most important variable is p_{max}, which plays an essential role in our proposed approach. It serves two purposes: It helps to indicate the progress of localization. For example, if $\mathrm{p_{max}}$ exceed a predefined threshold such as 80%, then the robot will stop to indicate it has been localized by itself and its pose is the representative of the considered

cluster. On the other hand, if p_{max} is equal to zero, it means there is no cluster exist anymore and the robot fails to localize itself. The second purpose is to be used as an indicator of the level of certainty used in the process of detection of robot discussed in 3.3.2. For instance, if $p_{A\_max} = 60\%$ and $p_{B\_max} = 25\%$, they represent the percentage of the largest cluster of robot A and B respectively. Then we consider that robot A has 60% of certainty about where it is, while robot B only has 25% of certainty about its location. By comparing the level of certainty of both detected robots, it could decide the direction of information exchange.

3.3 Proposed Approach

After introducing the clustering component, in this section, we present the proposed approach, which is the clustering based MCL for cooperative multi-robot localization algorithm.

3.3.1 Problem statements

As we mentioned before, our approach makes use of the information derived from clustering component for communication, and we could deal with the delayed information exchange problem. Here we state the strategy of how we tackle the problem.

The delayed information exchange problem means robots exchange their internal beliefs to update their pose estimates instantly whenever they detect each other, however, this schema could not ensure it will always benefit their pose estimates. It would be better to delay the information exchange especially when both detected robots are with higher uncertainty to their poses at the time of detection. In our approach, this problem comes in

two situations: The first one is that both detected robots are with blurry knowledge of their positions. This kind of situation always happens in the beginning of the localization due to both robots have not yet detected any landmarks for updating their internal beliefs (landmark for our experiments could be either an obstacle or a wall or a well localized robot). The second situation is that the detection happens right after one information exchange in the last detection, and within the period of time between two detections there is no detection of any landmarks by both robots. Since the internal beliefs have been exchanged at the last detection, if there is no detection of any landmarks the knowledge of its pose will stay in the same level, or become even worse due to the noises happened in robot's motion. In this consideration, the information exchange will not help the robots to refine their pose estimate. Therefore, our proposed approach tries to prevent the localization process from suffering the delayed information exchange problem.

In order to overcome this problem, one Boolean type status variable used to indicate the current state of landmark detection is needed for each robot, and the initial status of all robots will be set to false. By introducing this status variable, all robots can only have two states during the whole life of localization: The status variable value is false, which means there is no landmark has been detected by the considered robot yet or the robot has already exchanged its belief with another one. In the former case, the robot is always with highly uncertainty about its pose. The latter case is used to avoid the exchanging of same level of beliefs. The other situation is the status variable value is true, where a robot has already detected a landmark in the environment or the robot has already been localized. Detection of landmark will help a robot to refine its internal belief relative to the environment. Thus under this situation a robot will be more sure about where it is. When any robot has finished its localization process, the status variable will always be true, and then it will be used as a fixed landmark for other robot.

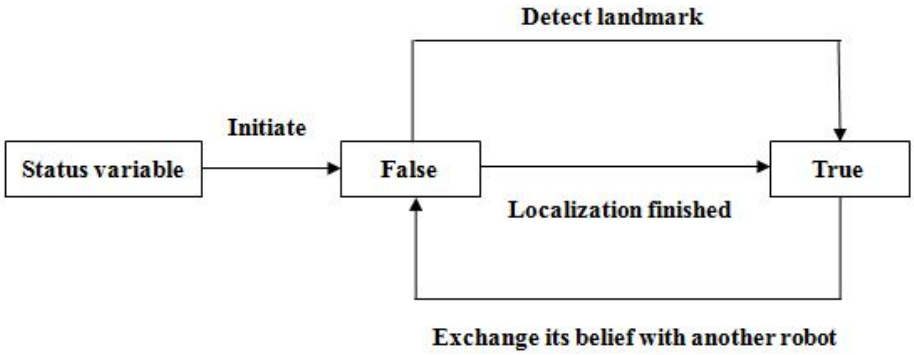


Figure 3.2: The Status variable diagram.

Figure 3.2 shows the relationships between two Boolean type values of status variable of each robot. Initially, each status variable is set to be “False”. During the whole time of localization, once a robot has detected any landmark, its status variable will be set to “True”, and once a robot has exchanged its belief with another robot, the status variable will be set to “False”. If one robot has finished its localization, the Boolean value will always stay in “True”.

With the help of status variable, robots are capable of monitoring their behaviour after detecting landmarks and other robots. Based on this capability, our strategy for dealing with the delayed information exchange problem manages the detection of two robots in two groups:

(a) In the first group, both status variables of two detected robots are false. This group includes two situations: The first one is both robots have not detected any landmarks yet. Here we can assume that the internal beliefs of both robots about their poses relative to the environment could be very blurry. Our approach just skips the information exchanging at current detection time since the blurry knowledge will not benefit the localization process or the contribution is too small and should be ignored. The

second situation is both robots have already exchanged their beliefs at last detection, and no landmarks have been observed by both robots yet at current detection time. As mentioned above, without a newly observation of landmarks the level of uncertainty will almost stay in the same level or even worse. Normally there is no need to exchange the same information again. Therefore our approach does not allow the information exchanging under this kind of situations.

(b) In the second group, there is at least one status variable value is true or both are true. Since perceiving a landmark helps robot to gather more information about its environment so that could help in its pose estimate, we consider it as necessary to exchange their internal beliefs to refine the pose estimate in both situations, which are only one robot has detected a landmark or both robots have detected landmarks. The idea of refining belief is to use the robot's belief which is with higher certainty about where it is to refine the other with lower knowledge about its pose and hence it requires a comparison of the level of certainties about their locations. In order to quantify the belief, our approach makes use of the variable p_{max} extracted from the clustering component. Through comparing of p_{max} of both robots we can find which one is with higher certainty about its position. And then our approach uses the information of the robot with higher certainty to help the one with lower certainty to refine its pose estimate. If they are with the same level of certainty, then we consider it as no need to exchange their information since usually same level of certainty will not produce useful results after refining.

Through this above strategy our approach could ensure the information exchange always benefit the localization process. Therefore our approach can increase the effectiveness and efficiency of multi-robot localization via an appropriate information exchange mechanism.

3.3.2 Details of proposed method

In this section before presenting our complete proposed approach, we firstly state the following assumptions:

(a) The number of robot is two, which is sufficient to test our proposed approach in both real robots experiment and simulation. Both robots are running on a 2-D flat platform. Each robot executes its own belief functions to model its own uncertainty.

(b) Each robot is equipped with the same sensors including wheel encoder sensors, bump sensors, and infrared sensors. All the collected information is used to update its own pose estimate.

(c) Each robot is capable of exchanging data with another via Bluetooth enabled device. The communication time is very short and can be ignored.

Our proposed approach written in pseudo code is shown in Table 3.2. The clustering component has been introduced into basic MCL algorithm. By analyzing the information extracted from clustering component, we propose a new information exchange mechanism for the communication between robots, which only allow an information exchange happen whenever it will contribute positively to the localization process.

Algorithm Clustering based MCL for multi-robot

Part 1:

Initially, each robot executes its own MCL+BSAS function to monitor its own pose estimate and cluster information.

If the degree of certainty p_{max} exceeds a predefined threshold η, the considering robot will stop to indicate it has been localized and return its location to be used as landmark for other robots.

Part 2:

When one robot detects another:

(a) If the values of both detected robots' status variables are false, there is no information exchange;

(b) Else there is one robot's status variable is true or both of them are true, then do belief comparison:

If they are not with the same degree of certainty, do information exchange which use the one with higher certainty to help the other one to refine its pose estimate;

Else there is no information exchange;

Part 3:

Process of information exchange:

(a) After determining the exchange direction, a predefined distance threshold γ is used to help in refining process:

Those clusters of the robot with lower certainty, whoever is within the range of predefined distance threshold γ to any clusters of the robot with higher certainty, will be kept;

(b) Return a new set of clusters which will focus on the more possible locations of the robot;

(c) Uniformly resample *N* total particles within the newly returned set of clusters;

Table 3.2: The clustering based MCL approach for cooperative multi-robot localization.

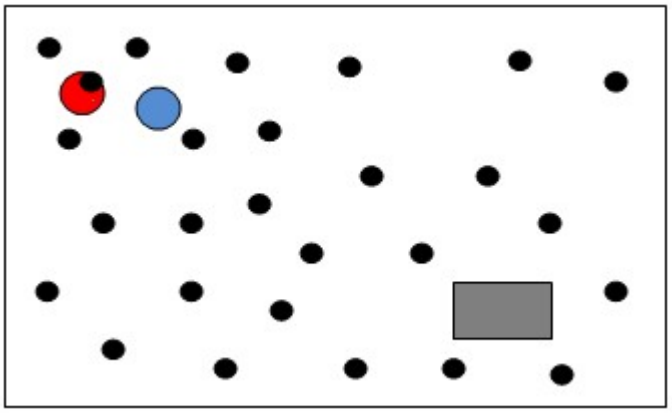
Our proposed approach consists of three parts. In part 1, each robot maintains its own belief function which is the MCL plus BSAS algorithm. The basic MCL is used to model its own uncertainty. The generated particle set is then applied to the BSAS algorithm which is the clustering component. By doing so, our approach could monitor

the localization process of each robot on real time, such as the number of clusters that the considered robot has, the locations of these clusters, the possibility of each cluster represent the true location of the robot. If the percentage of the cluster contained the largest number of particles which is p_{max} exceeds the predefined threshold η (two values are used in our experiments, 70% or 80%), then the considered robot will stop to indicate it has been well localized, and return the pose of representative of the cluster contained largest number of particles to be used as a landmark for the other robot. If p_{max} is equal to 0, which means there is no cluster exist at the current time, then the robot will stop to indicate it failed to localize itself. The ability that recognizing whether a robot has been well localized or not by itself significantly improves its robustness and autonomy.

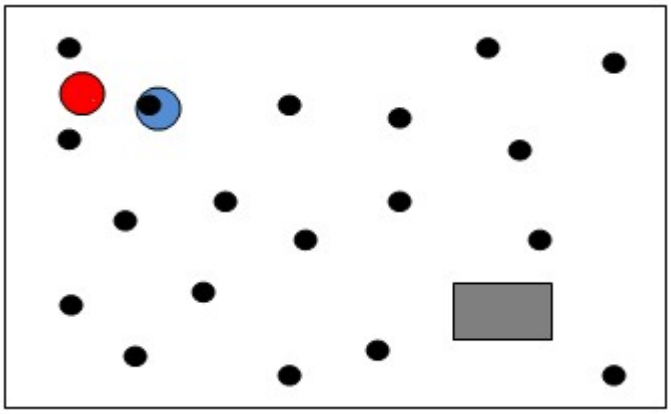
As we said the core idea of multi-robot localization is to let the robot benefit from information collected by others, therefore the interactions between robots is the essential part of our proposed approach. With the help of detection, robot can perceive other one, and then exchange their information with each other through Bluetooth enabled devices. In the absence of detection, each robot maintains its belief function independently specified in part 1. Now we come to the cooperative phase. Firstly in part 2 our proposed approach manages the behaviour of both detected robots in a principled way in order to dealing with the delayed information exchange problem. The behaviour is mainly to determine whether to allow an information exchange or not, and the direction of information exchange with the help of one important status variable described in the last section. In the following we give some examples to illustrate the two groups of the events while detection happens:

(a) In the first group, the values of both detected robots' status variables are false. Two situations will cause this happen. Normally the first situation occurs in the first time

of detection since both robots have not perceived any of the landmarks yet (the initial value of the status variable is set to be false). As we are dealing with global localization using MCL, all particles will be uniformly distributed all over the state space to model the global uncertainty. If there is no observation of any of landmarks to help robot to update its knowledge about the giving environment, the uncertainty will still stay in a relatively high level. Thus the exchange of their beliefs will not benefit the localization process. Figure 3.3 is a visual perception of this situation. Both robot A (red circle) and robot B (blue circle) are running on the same environment with a rectangle shape. Each black dot is a representative of one cluster which contains a certain number of particles. Figure 3.3(a) represents the current internal belief of robot A, while Figure 3.3(b) represents the internal belief of robot B at the same time. We can see if there is no detection of any landmarks, the beliefs of both robots are highly uncertain even though they move around for a while. At this level of uncertainty, the exchange of information will only increase the computational burden without any help to the localization process. Therefore we do not allow an exchange of information happen in this situation.



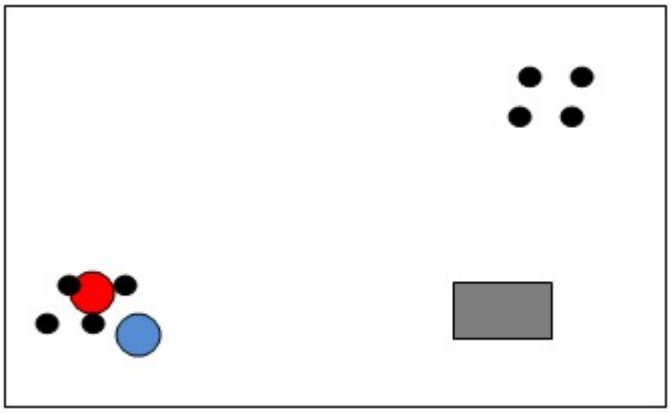
(a)



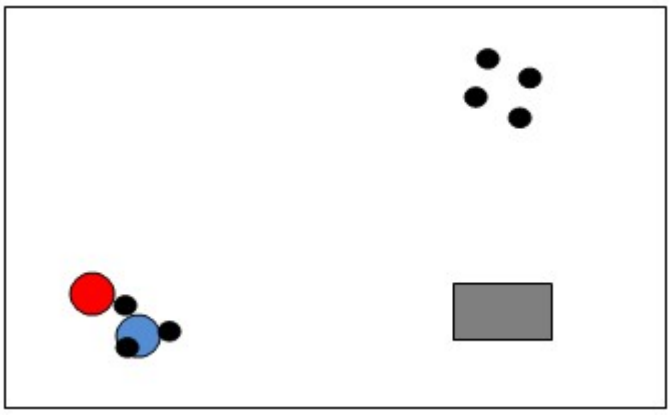
(b)

Figure 3.3: The first situation of detection without information exchange. (a) The current belief of robot A (red circle) represented by a set of clusters (small black dot); (b) The current belief of robot B (blue circle) represented by a set of clusters (small black dot). Both robots are running in the same environment and the gray rectangle is an obstacle.

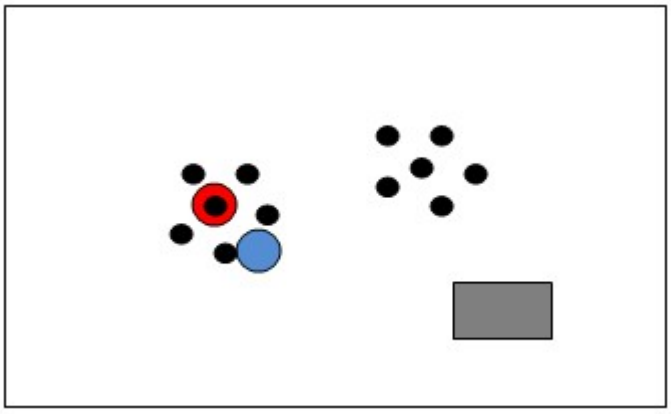
The second situation is that the current detection happens right after one information exchanging at last detection, and without any detection of landmarks by both robots within this time period. In this situation, the internal beliefs of both robots have been exchange in an early time. If no landmark has been perceived by one of them to update their internal beliefs, we consider it as unnecessary to exchange the same level or even worse of uncertainty. Figure 3.4 illustrates this situation in the same environment with previous situation. Figure 3.4(a) and (b) represent the beliefs of robot A and robot B respectively after an information exchange at the last detection. Both status variables have been set to false. After moving for a few seconds, they detect each other again without perceiving any landmarks. Figure 3.4(c) and (d) shows the current beliefs of robot A and robot B. As noises occur during the robots' motion, the beliefs of both robots become more and more uncertain. An information exchange in this situation will not further benefit the localization process than the last time of information exchange. Therefore, we consider it as no need to do the information exchange.



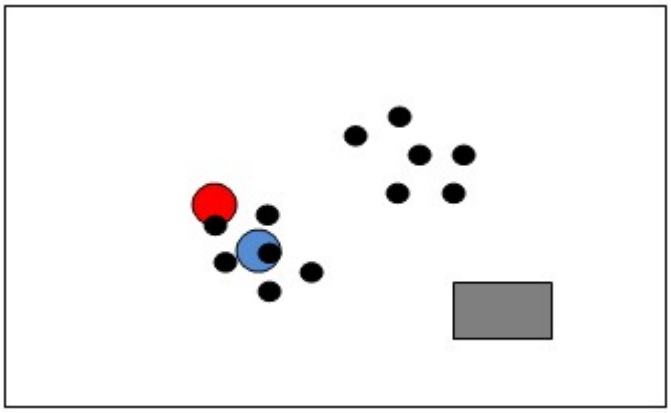
(a)



(b)



(c)



(d)

Figure 3.4: The second situation of detection without information exchange. (a) The belief of robot A (red circle) represented by a set of clusters (small black dot) after information exchanging at last detection; (b) The belief of robot B (blue circle) represented by a set of clusters (small black dot) after information exchanging at last detection. (c) The belief of robot A at current detection. (d) The belief of robot B at current detection.

(b) In the second group of detection, there is at least one of detected robots' status variable is true, or both of them are true. Since any detection of landmarks will help robot to refine its pose estimate, we consider it as necessary to exchange their information no matter it is the situation of one robot has detected landmarks or both robots have detected landmarks. As we described the strategy before, the idea of refining process is to use the belief of robot with higher certainty about where it is to help the other one with lower certainty. Then determining the direction of information exchange requires a way to evaluate the degree of certainty. Our approach use the information extracted from clustering component to quantify the degree of certainty. In this thesis, we only take p_{max}, the percentage of cluster that contains the largest number of particles, as the degree of certainty according to the major feature of the resampling phase of MCL algorithm described as the more number of particles fall in a region of space the more likely it is the true pose of the robot. Thus, p_{max} is sufficient to evaluate the degree of certainty, and then direction of information exchange could be easily determined by a comparison of p_{max}. For instance, p_{A_max} and p_{B_max} represent the uncertainty of robot A and B.

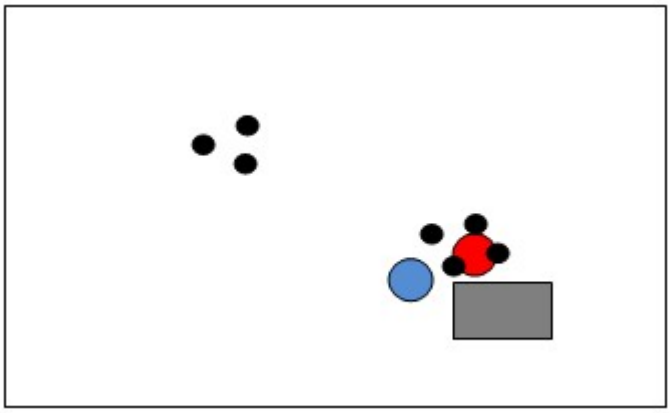
When detection happens, the direction of information exchange will be one of the following three cases:

1. If $p_{A\_max} > p_{B\_max}$, which means robot A is more certain about where it is than robot B, so robot A will use its belief to help robot B to refine its pose estimate;

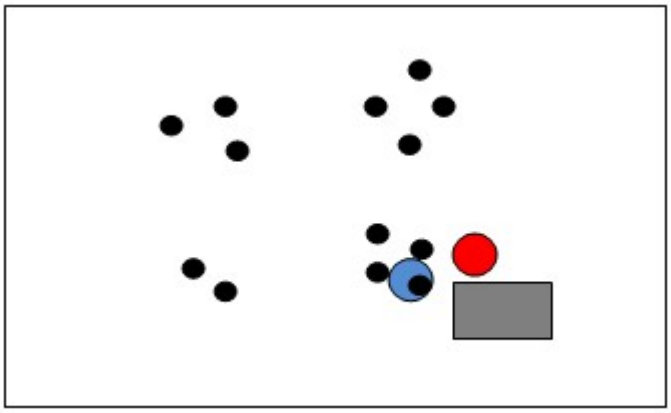
2. If $p_{B\_max} > p_{A\_max}$, which means robot B is more certain about where it is than robot A, so robot B will use its belief to help robot A to refine its pose estimate;

3. If $p_{A\_max} = p_{B\_max}$, which means robot A and robot B are with the same degree of certainty, there is no information exchange between robots. This situation however rarely happens.

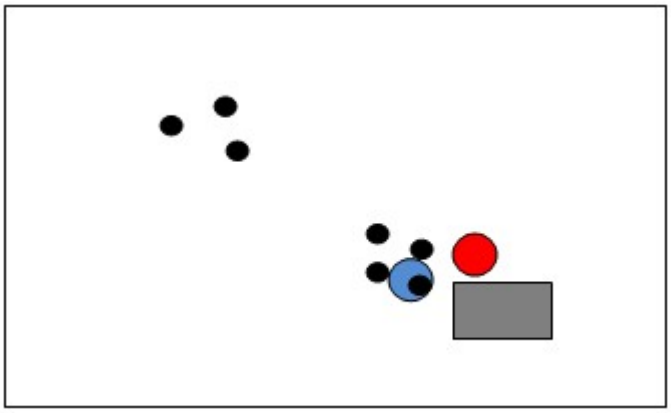
Figure 3.5 illustrates one example of detection under the situation of both robots have perceived some landmarks, and robot A is more certain about its pose than robot B. Figure 3.5(a) shows the current belief of robot A, and Figure 3.5(b) shows the current belief of robot B. Since this time of detection satisfies all constraints of information exchange, robot A uses its belief to refine the belief of robot B. Figure 3.5(c) shows the newly returned set of clusters of robot B after refining. The detailed process of information exchange will be given in the next paragraph. As a result, the localization process of robot B is accelerated.



(a)



(b)



(c)

Figure 3.5: One example of detection with information exchange. (a) The current belief of robot A (red circle) represented by a set of clusters (small black dot); (b) The current belief of robot B (blue circle) represented by a set of clusters (small black dot). (c) The new set of clusters of robot B after refining.

Here we discuss the part 3 of our proposed approach which is the detailed process of information exchange. As we mentioned above, each particle or cluster represents one possible position of robot. If two robots detect with each other, their estimated poses which are particles or the representatives of clusters should be within a certain distance. Our proposed approach makes use of this geometric relationship to build a mechanism for information exchange. Table 3.3 shows the pseudo code of the process of information exchange. It takes both sets of clusters of robot A and B, and the predefined threshold distance γ as inputs. The direction of information exchange has been determined by the rule that using the belief of robot with higher certainty to refine the one with lower certainty in part 2 of our proposed approach, thus, here we have two cases: Robot A refines robot B, if A is more certain about where it is (from line 1 to 10); robot B refines robot A, if B is more certain about its pose (from line 12 to 21). In case 1, line 2 initializes a new set of cluster C'_B. From line 3 to line 8, it is a for loop going through all clusters of robot B. Line 4 calculates the distance between the current cluster of robot B and all clusters of robot A to find the minimum one, if the minimum distance is smaller than the threshold γ, then add it to the set of cluster C'_B. Line 10 uniformly resamples the total number of particles within the newly returned set of cluster C'_B. The process of case 2 is the same with case 1 but with different direction of information exchange.

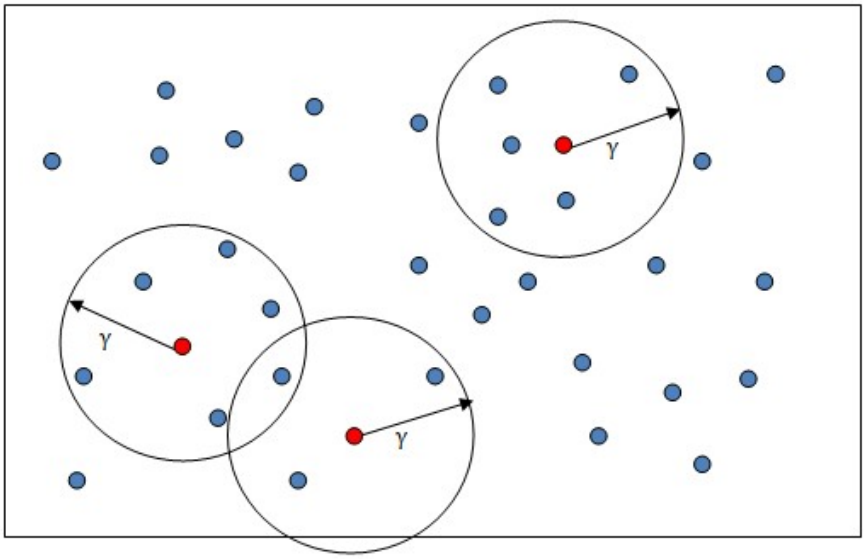
Information exchange ($\mathbf{C_A(a_1, \cdots, a_m)}, \boldsymbol{C_B(b_1, \cdots, b_n)}, \boldsymbol{\gamma}$)**:**

Case 1: Robot A refines Robot B
  $\mathrm{C}'_{\mathrm{B}} = \emptyset$
  for *i =1* to *n* do
    find $d(b_i, C_A) = min_{1 \le j \le m} d(x_i, C_j)$
      if $d(b_i, C_j) < \gamma$ then
        add b_i to C'_{B}
      endif
  endfor
  return C'_B
  uniformly resample particles within C'_B

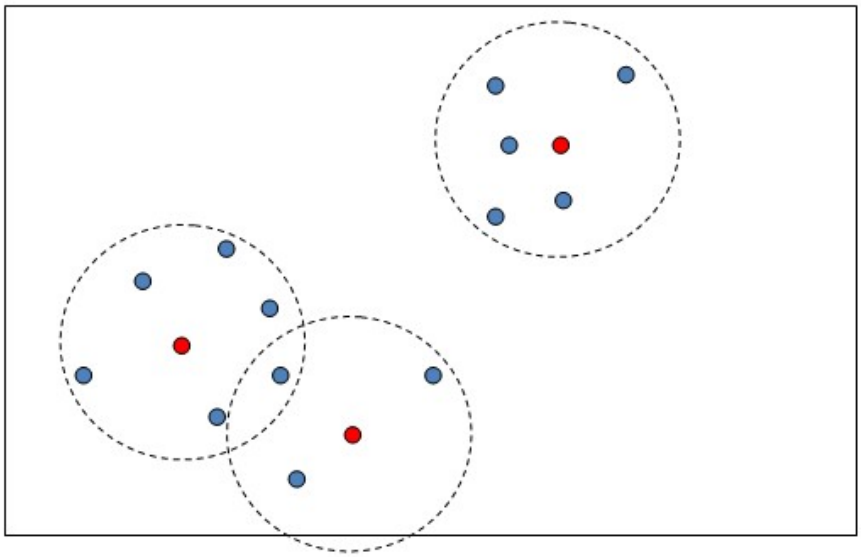
Case 2: Robot B refines Robot A
  $\mathrm{C}'_{\mathrm{A}} = \emptyset$
  for *i =1* to *m* do
    find $d(a_i, C_B) = min_{1 \le j \le n} d(x_i, C_j)$
      if $d(a_i, C_j) < \gamma$ then
        add a_i to C'_{A}
      endif
  endfor
  return C'_{A}
  uniformly resample particles within C'_{A}

Table 3.3: The process of information exchange.

Figure 3.6 is an example of information exchange with the direction of robot A refine robot B. Robot A and B are running on the same environment with a rectangular shape. Figure 3.6(a) shows the beliefs of both detected robots represented by two sets of clusters, red dots represent the clusters of robot A, while blue dots represent the clusters of robot B. The direction of current information exchange is robot A refines robot B. We can see the clusters of robot B, which are within the distance of threshold γ to any clusters of robot A, will be kept as reasonable pose estimate that can be detected by robot A. Figure 3.6(b) shows the newly returned set of clusters of robot B, now those clusters which are meaningless to the current detection have been deleted in order to achieve the goal of accelerating of localization process.



(a)



(b)

Figure 3.6: One example of information exchange with the direction of robot A refines robot B. (a) The current belief of robot A represented by a set of clusters (red dot), the current belief of robot B represented by a set of clusters (blue dot); (b) The newly returned set of clusters of robot B after information exchange.

One of the most important points in the process of information exchange is how to select an appropriate threshold as the geometric distance γ between two detected estimated poses (we use the representatives of clusters in our approach). The selection of

threshold γ highly depends on the setup of detection by robots. Either too large or too small of threshold γ will significantly affect the performance of localization. The larger number of threshold γ is, the less the robots will benefit from cluster refining since many meaningless estimated poses cannot be deleted. The effect of increasing the threshold γ will appear in the increasing time for localization. On the other hand, the smaller number of threshold γ is, the more likely it will remove some important clusters relative to the current detection so that will increase the failure rate. The selection of threshold γ for different experiments will be given in chapter 4 according to the type of detection technique.

CHAPTER 4

IMPLEMENTATION AND EXPERIMENTAL RESULTS

In this chapter, the details of the implementation of our experiment including hardware platform and software platform will be described firstly. Then the experiments results in both real and simulated environments will be presented.

4.1 Implementation Details

4.1.1 Hardware Platform

The hardware platform we used is the robot called iRobot Create which is an affordable mobile robot for educators, students and developers created by iRobot Corporation. The iRobot Corporation is founded from 1990. It has made some of the world's most important robots in two main fields: iRobot Home Robots and iRobot Government & Industrial Robots. In 2009 iRobot generated approximately $299 million in revenue and employed more than 500 of the robot industry's top professionals, including mechanical, electrical and software engineers and related support staff [12]. iRobot Create is preassembled and ready-to-use off the shelf, and relatively inexpensive. Another advantage is users can write custom software using a variety of methods that take advantage of the robot's "streaming sensor data" mode for more control of the robot, furthermore users can write programs for completely autonomous behaviour. The iRobot Create premium development package including Create programming robot, command module, advanced power battery system, virtual walls, standard remote and self charging

home base with fast charger only costs $299 and the Create programming robot only sells $129.

In this section, we will give the detail information about the Create in order to make our program to control it as comprehensive as possible. Three main sections structure the Create:

Sensor front: It includes all sensors equipped in Create such as bump, wall, cliff sensors, and home base contacts. All these sensors are mounted on the movable front bumper. This design is suitable for forward travelling. All sensors could easily sense any contact and absorb shock to minimize damage.

Motor middle: The drive motors and battery are designed in the center. The triangle location of wheels makes Create moving stably. The carry-on battery allows Create capable of long time running.

Cargo bay: The cargo bay connector is placed in the middle of the front part of the cargo bay. It contains 25 labelled pins that can allow users to attach electronics and other peripheral devices such as additional sensors, lights, or motors to the iRobot Create.

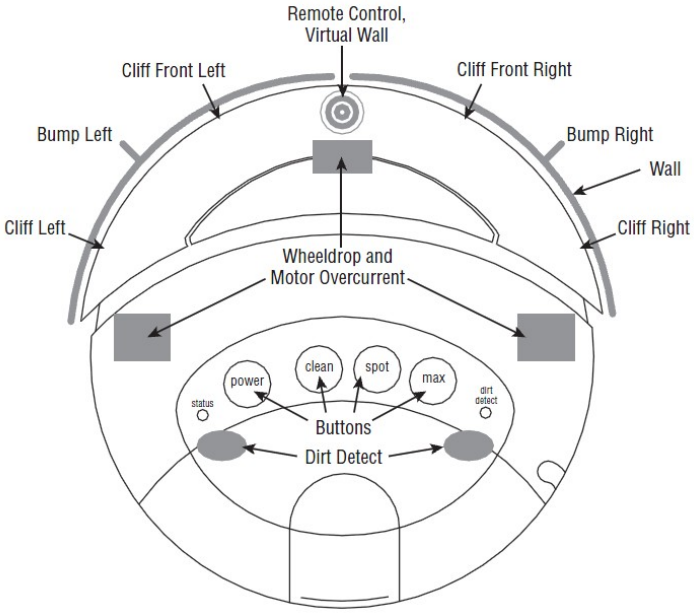


Figure 4.1: Location of Create sensors [23].

Sensors are very important components for an autonomous robot. All the locations of sensors of a Create is shown in Figure 4.1. For navigating, Create relies on its mechanical bump sensors, infrared wall sensors. For detecting dangerous conditions, it has infrared cliff detectors and wheel drop sensors. In our experiments we use three major groups of sensors:

Bump Sensors: There are two bump sensors in the front, located at 11 o'clock and 1 o'clock positions. When spring-loaded front bumper moves to activate one or both of these sensors, each sensor works as optical interrupter, which has a simple LED and photodetector pair: the LED shines and the photodetector detects the absence of the light and changes an electrical signal. In the case of Create's bump sensor, the interrupter is a small plastic arm connected to the bumper.

Infrared Sensors: The Create was equipped with six infrared sensors. All of them are located in the front bumper. Four of them are cliff sensors, which are facing down to the ground. Another one is wall sensor that faces to the right. These five sensors work

much like the bump sensor. Each sensor has one LED emitter and a photodetector looking for the LED's light. The difference is that they are detecting the reflected light of the LEDs other than detecting the direct light of those interrupt-based sensors. Cliff sensors are looking for the light reflected from the floor, and the wall sensor is looking for the light from walls. The last infrared sensor is the remote control/virtual wall/docking station sensor that is placed at 12 o'clock position on the front bumper. It is with a small round shape and has 360-degree lens that enables it to receive signals from any orientation. We use its capability of sensing virtual wall to perceive another robot in our experiments.

Internal Sensor: The three main internal sensors are wheel encoder sensors, wheel drop sensors, and power measurement sensors. The wheel encoder sensors provide us the information of travelling distance and rotating angle values by the optical interrupter sensor on the wheels. The distance value is calculated from counting the number of beam interruptions caused from the toothed interrupter disc. The angle value is obtained from an odometrical difference way. Each wheel has a directly distance sensor, and the angle value in the sensor data is calculated from the different travelled distance by each wheel. This difference describes a rotation around the center point between the two wheels. The wheel drop sensors are implemented with standard micro-switches, which are used to detect the state of the wheels. When the wheels are drops means Create is in some dire situation and should stop its current task. The power measurements sensors are used to estimate and present capacity of the battery.

Another important device of Create has to be mentioned is the BAM wireless accessory. The BAM wireless accessory provide us the capability of connecting and communicating with Create through Bluetooth enabled devices like computers. It's the most flexible method of communication. The effective connect range is 100 meters, and the connection quality decreases slowly over distance.

4.1.2 Programming Environment

The build-in serial protocol allowing us to connect and control a Create from a Bluetooth enabled devices is called Roomba Open Interface or ROI. It provides an electronic interface and a software interface that can control the Create's behaviour and monitor its sensors. The electronic interface of ROI is a DB-25 connector located in the Cargo Bay for Create used in our experiments, which is connecting hardware and electronic for sensors and actuators. The software interface provides us many useful commands, such as demo commands, driving commands, song commands, sensor commands, cargo bay connector commands, and scripting commands, to control Create's behaviour.

The Create's internals is almost transparent to the users through the ROI. It abstracts many useful functions to let the operating of Create become much easier. Much of the low-level hard work is bypassed via the use of ROI, instead those work such as dealing with the motors and sensors has been taken care of inside Create itself. It has five operating modes: Off, on, Passive, Safe and Full. Each mode reflects one kind of behaviour of Create and one kind of response to subsequent ROI commands.

Off: Create responds to no commands over the ROI. Either turning on the Power button or toggling the DD hardware line it will be in "On" mode.

On: In this mode, Create is awake and is able to operate via its button or remote control. Sending a START command is the only way it can out of this state.

Passive: A START command can let Create turn into this mode from "ON" mode. In this mode, no control of Create is allowed through ROI, but we could read the

information of sensors, and the buttons work as usual. This mode is used to monitor the initial state of Create that is going to its business. Usually sending any CONTROL command over ROI can make it enters the “Safe” mode.

Safe: Receiving CONTROL command from “Passive” mode or the SAFE command from “Full” mode. In this mode, Create can be full controlled over ROI with several build-in safety exceptions: any detection of cliff and wheel drops, or the charger is plugged in and powered. Encountering any of this safety conditions Create will stop and turn into “Passive” mode.

Full: Create will switch to “Full” mode through sending a FULL command in the “Safe” mode. In this mode the safety exceptions no longer exist anymore, we could get full control of Create. Sending a SAFE command could switch it into “Safe” mode.

Either a ROI command or external events can change the mode of the Create. The relationships of five ROI states are shown in Figure 4.2.

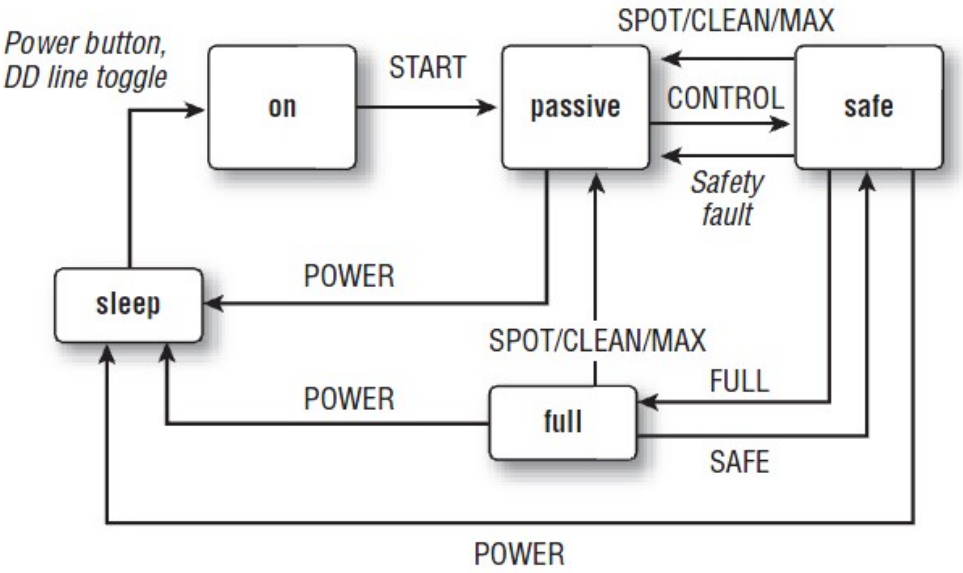


Figure 4.2: Create ROI state diagram [23].

All the ROI commands are varying number of data bytes. It is rudimentary and quickly gets tiresome. The RoombaComm API can help users to control Create a lot easier, which is an encapsulation of the ROI binary commands into a more easy-to-use Java class. It is a more human-readable code, and it can let users to create more complex behaviours of Create based on the “primitives” of ROI commands. The RoombaComm API is compatible with any operating system that RXTX (serial and parallel I/O libraries supporting Sun’s CommAPI) supports including Windows, Linux, and Mac OS [23].

The programming language we used to code is Java, and the environment we used to program is Eclipse. Java is currently one of the most popular programming languages, and it is a general-purpose, concurrent, class-based, object-oriented language that is specifically designed to have as few implementation dependencies as possible. Eclipse is a multi-language software development environment comprising an integrated development environment (IDE) and an extensible plug-in system. It allows users to write graphical programs, and is written mostly in Java and supports RoombaComm API and all other Java Class.

4.2 Experiments Results

Our approach will be tested on both real and simulated robots. In both real and simulated robot experiments, we firstly tested the performance of single mobile robot localization, and then applied our approach to multi-robot localization. Three important results: the time for localization t(s), the successful rate ε, and the error distance d, will be collected in every experiment. Finally we compare these characteristic results of multi-robot localization with single robot localization, and aim to verify that how much our approach can improve the localization quality.

4.2.1 Experiments using real robot

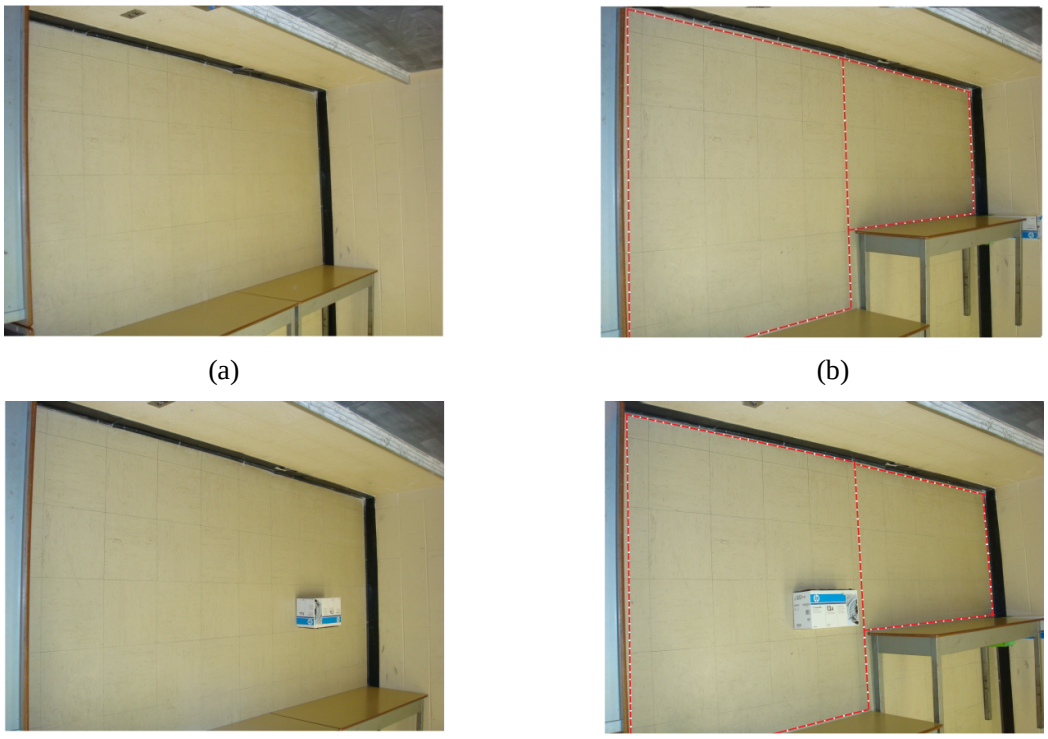
In the real robot experiments, two iRobot Create are used to test our proposed approach shown in Figure 4.3. We firstly implement the case of single mobile robot localization in four different environments. And then we implement the case of multi-robot localization using proposed approach in the same environments used in single mobile robot localization. The comparison of the results of these two cases will be shown and to see whether our proposed approach can benefit the localization process.

Figure 4.3: Two iRobot Create using in our experiments. The left Create is with black label, while the right one is regular Create.

(a) Experiments of Single mobile Robot Localization

The four different environments for single mobile robot localization are shown in Figure 4.4. These environments are featured as follow: symmetric environment (Figure 4.4(a)), asymmetric environment (Figure 4.4(b)), symmetric environment with obstacle (Figure 4.4(c)), and asymmetric environment with obstacle (Figure 4.4 (d)). We place one Create in each environment to perform global localization by using MCL+BSAS framework. The path of Create is set to turn left 120° when it bumps to the wall or

obstacle, otherwise keep going forward until the localization is finished. The total number of particles used to represent the belief of Create is 5000. The dissimilarity measure threshold θ is set to 17cm which is the radius of Create. Therefore the shape of each cluster will be the same size as Create. Threshold η is set to 70%, which means if the number of particles contained in the largest cluster is equal or larger than 70% of total, the Create will stop to indicate it has been localized. We repeat each experiment 50 times to track three important characteristic results: the localization time t(s), the successful rate ε, and the error distance d(cm) when localization succeeded(in real robot experiments, we consider localization is successful if the distance between the final pose of Create and the result pose of localization process is smaller than the diameter of Create 34cm).



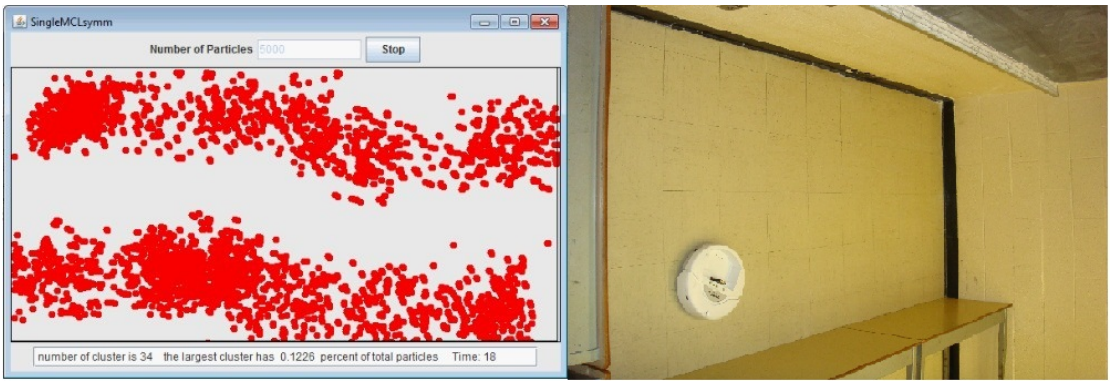
(a) (b)

(c) (d)

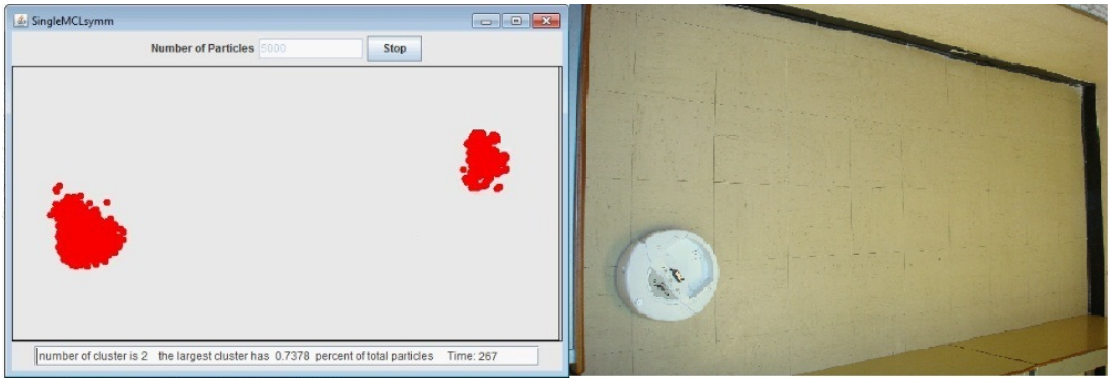
Figure 4.4: Four experimental environments of single mobile robot localization. (a) A rectangle field with the shape of 300cm×150cm; (b) Two rectangle fields, the shape of the left one is 150cm×150cm, the shape of the right one is 150cm×100cm; (c) A rectangle field with the shape of 300cm×150cm, and an obstacle with the shape of 31.5cm×19.5cm placed in the field; (d) Two rectangle fields, the shape of the left one is 150cm×150cm, the shape of the right one is 150cm×100cm, and an obstacle with the shape of 31.5cm×19.5cm placed in the field.

Figure 4.5 shows an example of single mobile robot localization tested in the symmetric environment (Figure 4.4(a)) with size 300cm×150cm. Normally the traditional MCL algorithm will fail to localize itself in this kind of environment due to particles will keep concentrating on some symmetric positions, and there is barely chance to solve the problem of geometric symmetry by perceiving information from this kind of external environment. However, the MCL+BSAS framework have a chance to deal with it through analyzing the number of particles contained by each cluster. Although particles may concentrate on some symmetric positions, the largest cluster still has a chance to reach the predefined threshold η. Then the representative of the largest cluster will be the estimated position of Create. Sometimes it will fail as well in indicating a false position to the Create. In this experiment Create starts with global uncertainty, and all particles are spread all over the map according to the uniformly distribution shown in Figure 4.5(a). After running 18 seconds, Figure 4.5(b) shows all particles spread on two large portions due to geometric symmetry. In Figure 4.5(c), the symmetric problem still has not been solved as all particles concentrate on two positions. However, the largest cluster reaches 73.78% at the time of 267 second, which is larger than the threshold η, the process stops and indicates Create has been localized in the position of the pose of representative of the largest cluster. In this example, the localization is successful as the distance between the final pose of Create and the pose of the largest cluster is smaller than 34cm.

(a)



(b)



(c)

Figure 4.5: Example of single mobile robot localization in symmetric environment. (a) Globally uncertainty, all particles are uniformly distributed at the beginning; (b) After 18 seconds, particles are spread on two large portions due to geometric symmetry; (c) Finally the largest cluster reaches the threshold η and Create is localized at the pose of the representative of the largest cluster.

Table 4.1 shows all the results of single mobile robot localization using iRobot Create in four different environments. We can see the longest average time taking for localization is 230.88 seconds, which is case of the Create localizing itself in symmetric environment. The problem of geometric symmetry not only affects the localization time, but also brings the lowest successful rate, which is 42%, due to it may fail in indicating a false position. This problem is easily solved in the other three environments which have features to break the geometric symmetry. The best results show in the case of asymmetric environment with obstacle since it has the most distinguishable features to

help the Create to localize itself. The average localization time is only 74.38 seconds, approximately one third of the longest one, and the successful rate reaches 90%, which is more than two times of the worst case. The cases of localization in asymmetric environment and symmetric environment with obstacle also shorten the localization time and increase the successful rate in a large extent than the symmetric one, which are: 81.6 seconds and 84%, 92.62 seconds and 80%, respectively. As to the average error distance, calculated by the distance between the true position of Create and the pose of representative of the largest cluster, the results for each case are: 12.86cm, 13.93cm, 13.4cm, and 14.11cm. They are all smaller than the radius of Create which means those localizations are successful.

Table 4.1: Results of experiments of single mobile robot localization using Create in four environments.

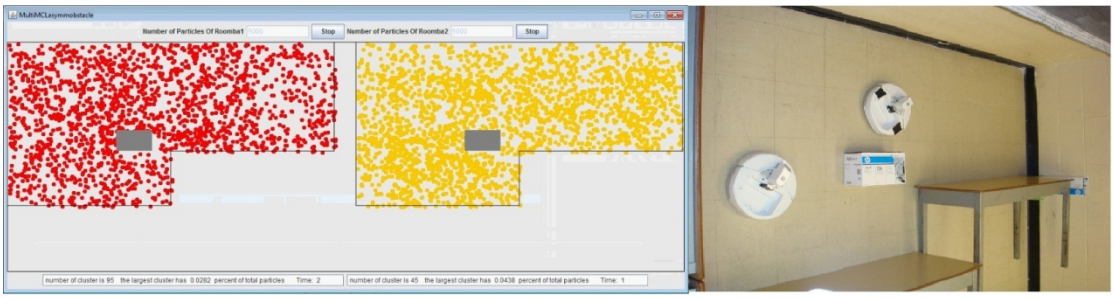
Single mobile robot localization using Create (□=70%)			
Environment	**Average localization time t(s)**	**Successful rate ε**	**Average error distance d(cm)**
Symmetric	230.88	42%	12.86
Asymmetric	81.6	84%	13.93
Symmetric with obstacle	92.62	80%	13.4
Asymmetric with obstacle	74.38	90%	14.11

(b) Experiments of Cooperative Multi-Robot Localization

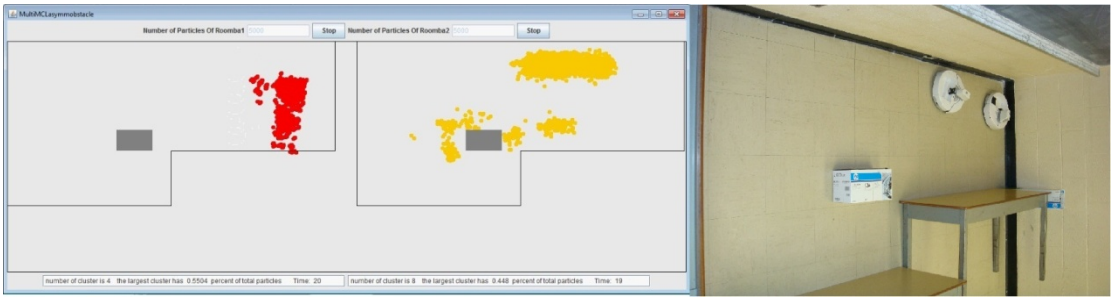
In this case of experiments, two Create are used to test our proposed approach in the same environments shown in Figure 4.4. These two Create perform global localization and they are capable of exchanging information with each other to refine their pose estimate when detection happens. The total number of particles for each Create is also

5000. The threshold γ used in the process of information exchange is set to be 60cm since the setup of detection is determined by the range of infrared signals emitted by Virtual wall sensor. The other predefined thresholds are the same with the previous experiments: dissimilarity threshold θ is set to be 17cm, and threshold η is also set to 70%. The path of both Create is set to turn left 120° when it bumps to the wall or obstacle or another Create, otherwise keep going forward until the localization is finished. We repeat each experiment 50 times to track three characteristic results: the localization time t(s), the successful rate ε, and the error distance d(cm) when localization succeeded.

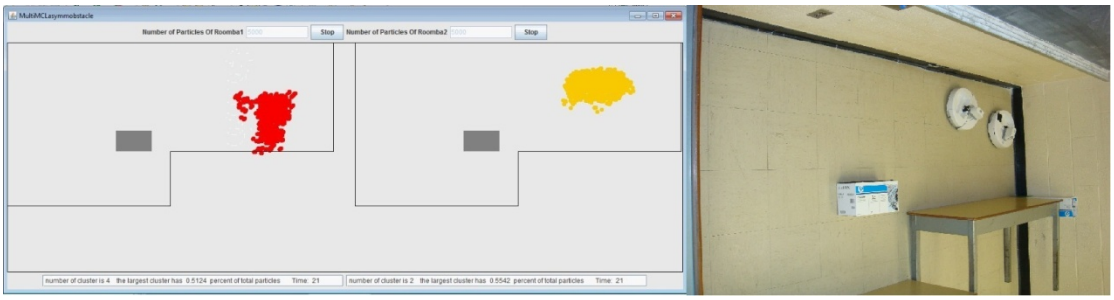
Figure 4.6 shows an example of cooperative multi-robot localization in asymmetric environments with an obstacle. The red particle set in the left part of the window represents the belief of the Create with black label, while the yellow particle set in the right part of the window represents the belief of the other regular Create. Figure 4.6(a) shows the global uncertainty represented by uniformly distributed particles at the beginning. Detection happens in Figure 4.6(b), the Create with black label is more certain about where it is since the largest cluster contains 55% of total particles which is higher than the other one 44.8%. Both Create have just bumped to the wall and updated their beliefs about its external environment. According to our proposed approach they satisfy the constraints to exchange their information to refine the regular Create's belief. The detailed refining process is given in chapter 3. Figure 4.6(c) shows a new particle set of the regular Create after refining. We can see the cluster number has been reduced from 8 to 2, and all particles focus on the more possible position of the regular Create. Therefore, our proposed method can achieve the goal of accelerating the localization process. Figure 4.6(d) demonstrates that both Create have localized themselves successfully through our proposed approach.



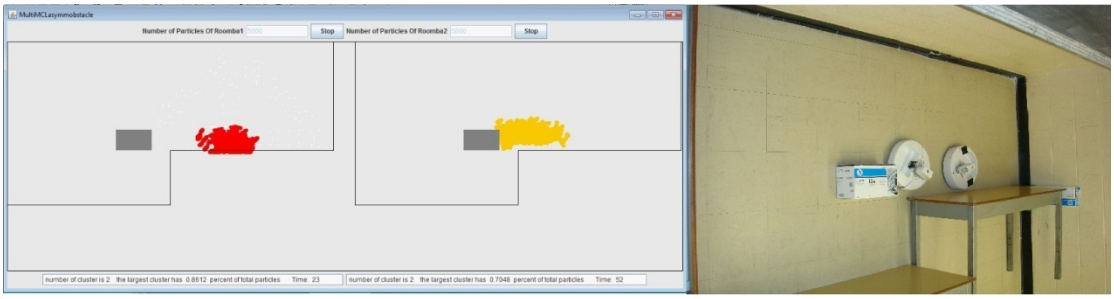
(a)



(b)



(c)



(d)

Figure 4.6: Example of cooperative multi-robot localization in asymmetric environment with obstacle. Red particles represent the belief of Create with black label, while the yellow particles represent the belief of regular Create. (a) Globally uncertainty, two particle sets are uniformly distributed at the beginning; (b) One detection happens, the belief of Create with black label is more certain than the regular one; (c) The Create with black label uses its belief to help the regular one to refine its belief; (d) Finally, both Create have successfully localized themselves.

The results of all cases of cooperative multi-robot localization using two Create are shown in Table 4.2. All three tracing results ($t, \varepsilon, and\ d$) here are the average number of both Create for each case. The average localization time in symmetric environment is 144.04s, while others are only 60.73s, 69.54s, and 51.76s in asymmetric environment, symmetric environment with obstacle and asymmetric environment with obstacle respectively. The successful rate for each environment is 72%, 86%, 84%, and 91%. These results also illustrate that the more symmetric the environment is, the longer time they need to localize themselves, and the easier they might fail to localize themselves. As to the average error distance d here, 13.3cm, 14.585cm, 14.35cm and 15.075cm for each environment, all of them are even smaller than the radius of Create. All these tracing results of multi-robot localization illustrate that our proposed approach could work very well for cooperative multi-robot system.

Table 4.2: Results of experiments of cooperative multi-robot localization using Create in four environments.

Multi-robot localization using Create (□=70%)			
Environment	**Average localization time t(s)**	**Successful rate ε**	**Average error distance d(cm)**
Symmetric	144.04	72%	13.3
Asymmetric	60.73	86%	14.585
Symmetric with obstacle	69.54	84%	14.35
Asymmetric with obstacle	51.76	91%	15.075

Table 4.3 summarizes the comparison results of multi-robot localization with single robot localization by using two Create. The results demonstrate that compare to the experiment of single robot localization, our proposed approach applied in multi-robot localization not only reduces the time for localization, but also increase the successful rate

for each robot. We can see without the help of detection of other Create and exchange information between them the time for single robot localization in symmetric environment is 230.88s, and the successful rate is only 42%. With the help of another Create, the results show significant improvement, which reduce the time for localization by 37.6% to 144.04s in average of two Create, and increase the successful rate by 30% to 72%. The large improvement of time saving also shows in the other three environments, which are 25.6%, 24.9%, and 30.4% respectively. However our proposed approach can only increase the successful rates by relatively small ranges, which are 2%, 4%, and 1% in asymmetric environment, symmetric environment with obstacle, and asymmetric environment with obstacle respectively. This is because these three environments are very distinctive and can easily deal with geometric symmetry problem, and single Create can also achieve high successful rate. In summary, our proposed approach can significantly shorten the time for localization in both symmetric environment and featured environments compare with single robot localization. As to the successful rate, it depends on the type of environments. The improvement is obviously while localizing in symmetric environments.

Table 4.3: Comparison Multi-robot localization with Single robot localization by using Create.

Comparison Multi-robot localization with Single robot localization using Create					
Environment	**□ =70%**	**Average localization time(s)**	**Time saving (%)**	**Successful rate ε**	**Increasing of successful rate**
Symmetric	**Single robot**	230.88	37.6%	42%	30%
	Multi-robot	144.04		72%	
Asymmetric	**Single robot**	81.6	25.6%	84%	2%
	Multi-robot	60.73		86%	
Symmetric with obstacle	**Single robot**	92.62	24.9%	80%	4%
	Multi-robot	69.54		84%	
Asymmetric with obstacle	**Single robot**	74.38	30.4%	90%	1%
	Multi-robot	51.76		91%	

4.2.2 Simulation Experiments

The simulation experiments are also implemented in four different environments: symmetric environment, asymmetric environment, symmetric environment with external obstacle, and asymmetric environment with external obstacle. Both single mobile robot localization and multi-robot localization have been tested under these four environments. Then we will further study the parameter η, and we try different values to see how it can affect the results of localization. As to the other two important parameters θ and γ, they tend to be determined by the hardware using in the experiment, thus we only interest in testing parameter η.

(a) Experiments of Single mobile Robot Localization

Single robot global localization using MCL+BSAS framework is implemented in four different simulated environments shown in Figure 4.7. The total number of particles used to represent the belief of each robot is 5000. The dissimilarity measure threshold θ is set to be 60 pixels (3 times of the radius of robot), and threshold η is set to 70%. We repeat each experiment 50 times to track three important characteristic results: the localization time t(s), the successful rate ε, and the error distance d(pixel) when localization succeeded(in simulated experiments, we consider localization is successful if the distance between the final pose of Create and the result pose of localization process is smaller than 40pixel).

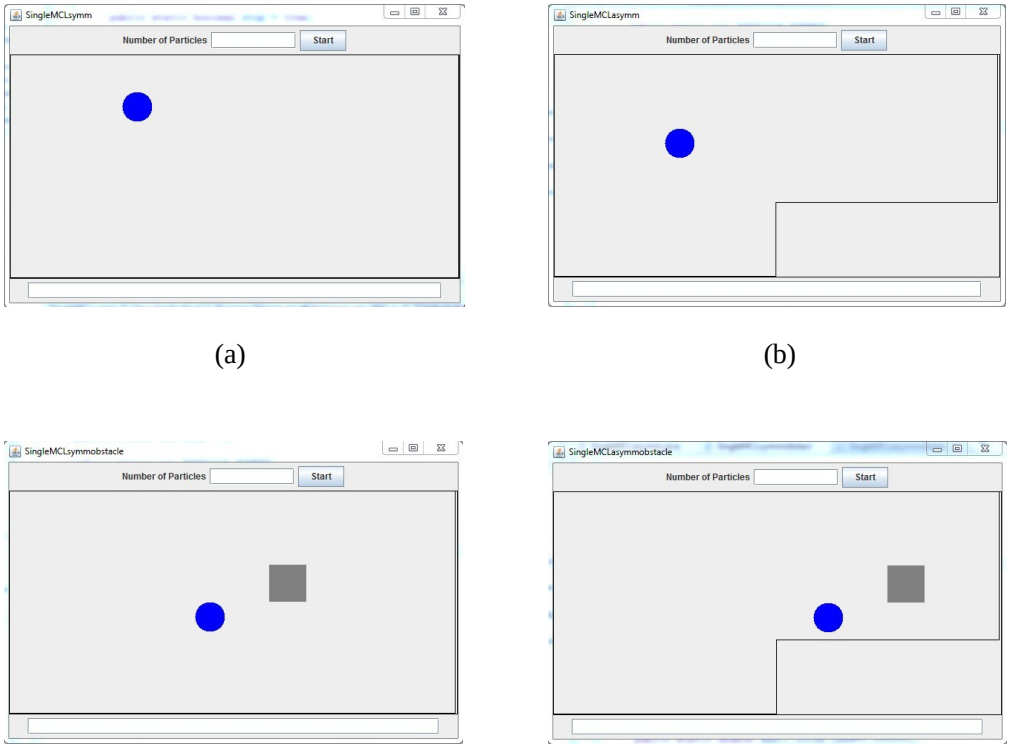
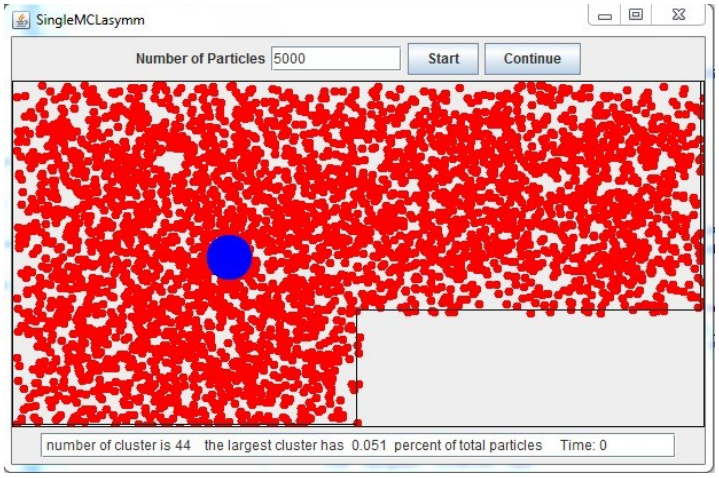


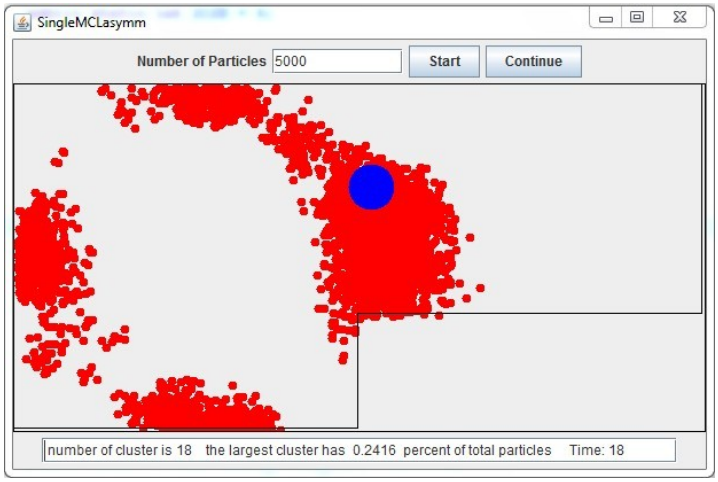
Figure 4.7: Four simulation environments of single mobile robot localization. (a) A rectangle field with the shape of 600×300pixel; (b) Two rectangle fields, the shape of the left one is 300×300pixel, the shape of the right one is 300×200pixel; (c) A rectangle field with the shape of 600×300pixel, and a gray rectangle obstacle with the shape of 50×50pixel placed in the field; (d) Two rectangle fields, the shape of the left one is 300×300pixel, the shape of the right one is 300×200pixel, and a gray rectangle obstacle with the shape of 50×50pixel placed in the field. The blue circle represents the robot.

The four environments vary either in the shape or the external obstacles. In Figure 4.7(a), the environment is totally symmetric. It could be assumed that it is the most challenging one for single robot system to locate itself. On the other hand, the environments of Figure 4.7(b), (c), (d) have the characteristics of, asymmetric shape, external obstacle, and asymmetric shape with external obstacle, respectively. These different features would help the robot to locate itself easily and quickly. In each environment, a robot is to locate itself starting from globally uncertainty. When robot hit the wall or obstacle, it will turn left 120°, otherwise go forward until localization finished. We captured one example of the global localization of single mobile robot in the

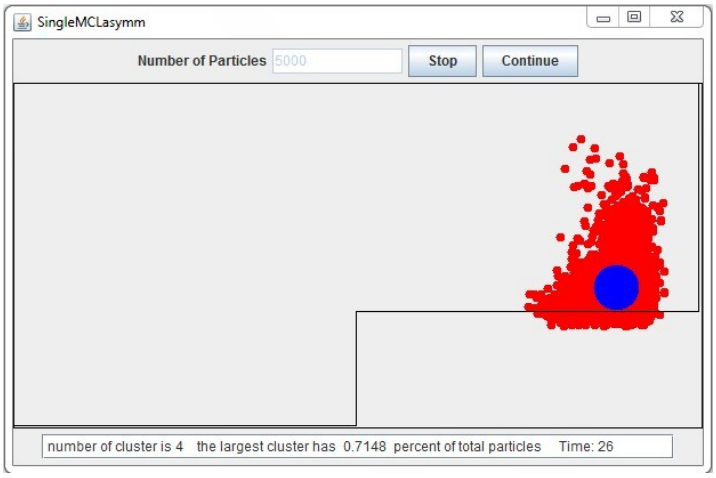
environment of Figure 4.7(b) as an example shown in Figure 4.8. This asymmetric environment consists of a left rectangle with size 300×300 pixel and a right rectangle with size 300×200 pixel. In Figure 4.8(a), at the beginning all particles are uniformly distributed all over the environment due to the global uncertainty. The clustering results are shown in the text box. At current time, there are 44 clusters, and the largest cluster contains 5.1% particles of total. After 18 seconds, Figure 4.8(b) shows that the robot is more certain about where it is since most particles are concentrated on four possible positions. The clustering information shows 18 clusters exist, and the largest cluster has 24.16% particles of total. Since p_{max} is still lower than η, the localization process will keep going until it reach the threshold. Finally, at the time of 26 seconds, almost all particles are centered on the true position of robot in Figure 4.8(c). We can see p_{max} is 71.48% which is higher than the threshold 70%, and only 4 clusters left at this time. The robot will believe it has been successfully localized and the localization process should stop. The representative of the largest cluster indicates the location of the robot.



(a)



(b)



(c)

Figure 4.8: Example of simulation experiment of single mobile robot localization in asymmetric environment. (a) Globally uncertainty, all particles are uniformly distributed at the beginning; (b) After 18 seconds, particles are concentrated on four possible positions; (c) Finally all particles are concentrated on the true position of the robot.

The tracing results of experiments under four different environments are shown in Table 4.4, included average localization time t(s), the successful rate ε, and average error distance d(pixel) when localization succeeded. We try 50 times of each experiment and take the average result under threshold η is 70%. As we can see from Table 4.4, the worst results show in the experiments under symmetric environments, compared with the other three environments: asymmetric environment, symmetric environment with obstacle,

and asymmetric environment with obstacle. The average localization time is 368.02 seconds in the symmetric environment while others are only 31.4s, 33.24s, and 28.74s, respectively. The reason of the extremely long time for localization in the symmetric environment is all particles will concentrate on many areas which have the feature of geometric symmetry. Therefore, the largest cluster is hard to reach the threshold 70%. Not only the time for localization is almost 14 times of others, but also the successful rate ε is extremely low, which is only 44%. Most of the error cases showed that the localization failed in indicating the false pose of the robot. The chance of the largest cluster being the one in the symmetric position is very high in symmetric environment, however, this problem has been easily minimized in featured environments, as we can see the successful rate ε are 86% for asymmetric one, 84% for the one of symmetric with obstacle, and the highest number 92% for the one of asymmetric with obstacle. Another important criterion of our method is the error distance. As we can see the average error distance d for experiments in these four environments are 8.95 pixels, 10.23 pixels, 12.29 pixels, and 12.78 pixels. All of them are smaller than the radius of robot, and compare to the size of environments these results demonstrate that our approach could locate the robot within a small error distance.

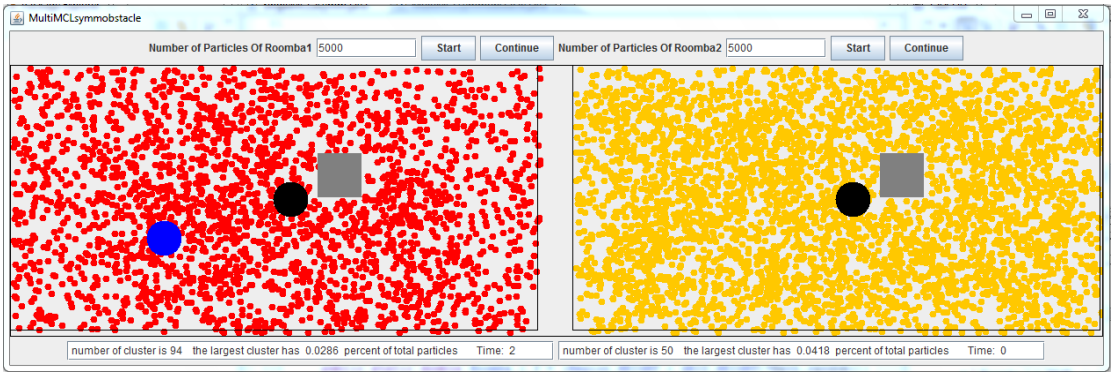
Table 4.4: Results of simulated experiments of single mobile robot localization in four environments.

Simulated Single mobile robot localization (□=70%)			
Environment	**Average localization time t(s)**	**Successful rate ε**	**Average error distance d(pixel)**
Symmetric	368.02	44%	8.95
Asymmetric	31.4	86%	10.23
Symmetric with obstacle	33.24	84%	12.29
Asymmetric with obstacle	28.74	92%	12.78

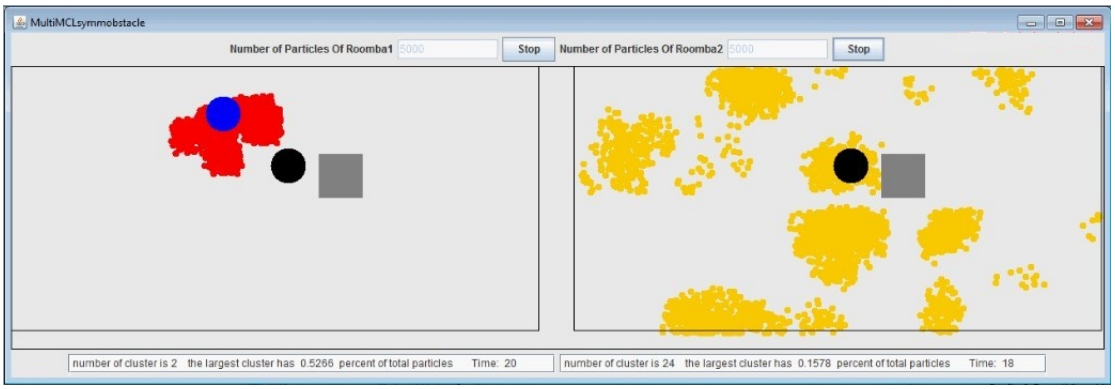
(b) Experiments of Cooperative Multi-Robot Localization

In this group of simulation experiments, the proposed approach is implemented in the same environments shown in Figure 4.7, but two robots are used together to perform global localization instead of single robot. The total number of particles is also 5000. The threshold is set to 120 pixels which is the same setup of detection with experiments using Create. The other predefined thresholds are the same with the previous single robot localization experiments: dissimilarity threshold is set to be 60 pixels, and threshold is also set to 70%. We repeat 50 times to track three characteristic results: the localization time (s), the successful rate , and the error distance (pixel) when localization succeeded.

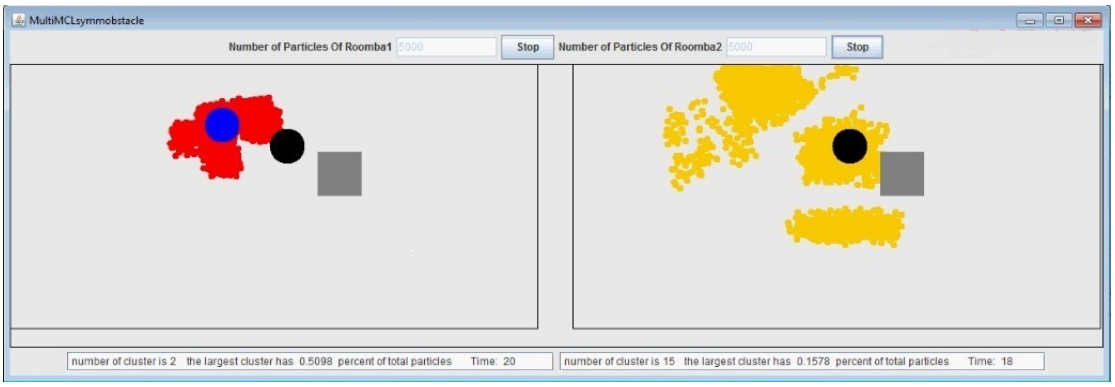
Since the processes of localization in the four different shape of environments are very similar to each other, we here only capture one typical run of the localization in the symmetric environment with obstacle which is the environment shown in Figure 4.9. This environment is with the rectangle shape of 600 300 pixel and a rectangle obstacle 50 50 pixel placing in it.



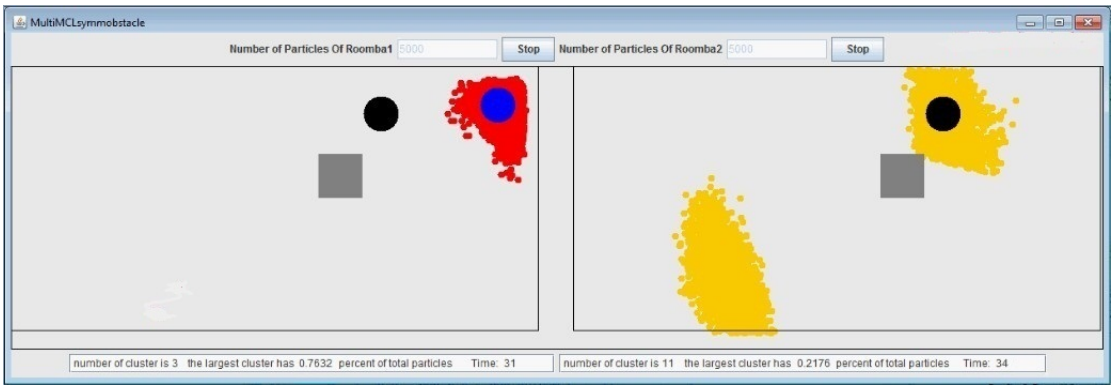
(a)


MultiMCLsymmobstacle
Number of Particles Of Roomba1
5000
Stop
Number of Particles Of Roomba2
5000
Stop
number of cluster is 2 the largest cluster has 0.5266 percent of total particles Time: 20
number of cluster is 24 the largest cluster has 0.1578 percent of total particles Time: 18

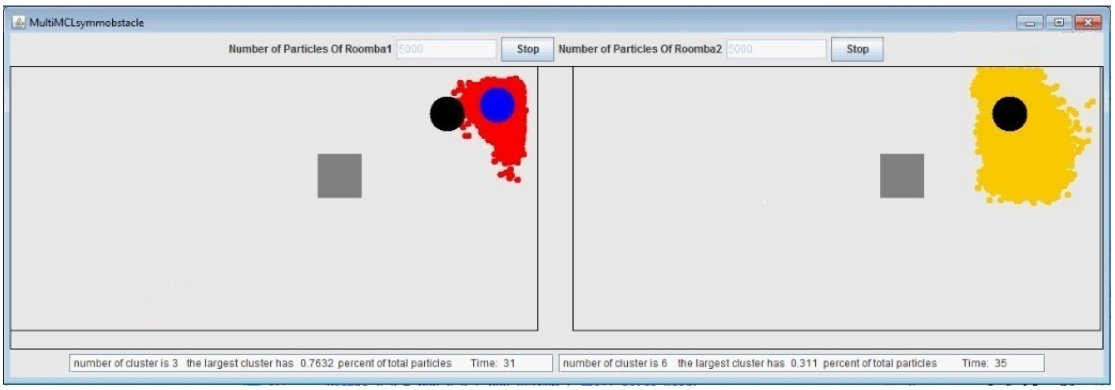
(b)


MultiMCLsymmobstacle
Number of Particles Of Roomba1
5000
Stop
Number of Particles Of Roomba2
5000
Stop
number of cluster is 2 the largest cluster has 0.5098 percent of total particles Time: 20
number of cluster is 15 the largest cluster has 0.1578 percent of total particles Time: 18

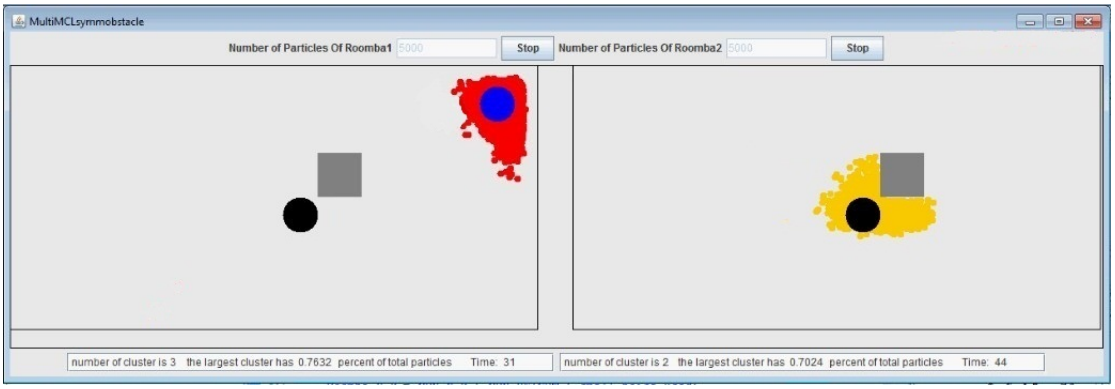
(c)


MultiMCLsymmobstacle
Number of Particles Of Roomba1
5000
Stop
Number of Particles Of Roomba2
5000
Stop
number of cluster is 3 the largest cluster has 0.7632 percent of total particles Time: 31
number of cluster is 11 the largest cluster has 0.2176 percent of total particles Time: 34

(d)



(e)



(f)

Figure 4.9: Example of simulation experiment of multi-robot localization in symmetric environment with obstacle. (a) Globally uncertainty, two particle sets are uniformly distributed at the beginning; (b) One detection happens, the belief of robot in blue is more certain than the black one; (c) The robot in black uses the belief of blue one to refine its belief; (d) Another detection happens when robot in blue is already localized; (e) The robot in black uses the belief of blue one to refine its belief again; (f) Finally, robot in black is localized.

Two robots (blue and black) are used to perform global localization in the same environment. They are capable of exchanging their beliefs, which are their clustering information, to benefit the localization process. Figure 4.9(a) shows the global uncertainty at the beginning. Two particle sets are spread over the environment (red spots represent the belief of robot in blue, and yellow spots represent the belief of robot in black). In Figure 4.9(b), two robots are within a certain range that can be detected with each other. At this time, the robot in blue is more certain about where it is, since the clustering results

show that two clusters exist in the current time, and the largest cluster contains 52% particles of total, which is higher than the black one. The largest cluster of robot in black only has 15% particles of total, and the total number of clusters is 24. And both of them have just detected the wall to update their beliefs which satisfy the constraints to exchange their clustering information. Therefore the belief of robot in blue is exchanged to the black one to help it refine its pose estimation. Figure 4.9(c) shows the refining results. As we can see now only 15 clusters left instead of the previous 24 clusters. And these 15 clusters are centred on the positions with relatively higher possibility that can represent the true location of the robot. Figure 4.9(d) and (e) show another case of detection. This time robot in blue has already been localized. It is used as an obstacle to help robot in black to refine its belief. We can see the geometric symmetry is successfully solved in this current detection. Figure 4.9(f) shows both robots have successfully localized themselves through proposed approach.

Table 4.5 shows the tracing results of multi-robot's experiments in four environments shown in Figure 4.7. All three tracing results ($t, \varepsilon, and\ d$) here are the average of both robots. The average localization time in symmetric environment is 109.56s, while others are only 22.485s, 24.25s, and 19.65s in asymmetric environment, symmetric environment with obstacle and asymmetric environment with obstacle respectively. The successful rate for each environment is 76%, 89%, 88%, and 93%. These results also illustrate that the more symmetric the environment is, the longer time they need to localize themselves, and the easier they might fail to localize themselves. As to the average error distance d here, 9.925 pixels, 11.975 pixels, 13.06 pixels and 14.7 pixels for each environment, all of them are smaller than the radius of robot. All these tracing results of multi-robot's simulation illustrate that our proposed approach could work very well for cooperative multi-robot system.

Table 4.5: Results of simulated experiments of multi-robot localization in four environments.

Simulated Multi-robot localization (□=70%)			
Environment	**Average localization time of two robots t(s)**	**Average Successful rate ε**	**Average error distance of two robots d(pixel)**
Symmetric	109.56	76%	9.925
Asymmetric	22.485	89%	11.975
Symmetric with obstacle	24.25	88%	13.06
Asymmetric with obstacle	19.65	93%	14.7

Table 4.6: Comparison Multi-robot localization with Single robot localization in simulated experiments.

Comparison Multi-robot localization with Single robot localization					
Environment	**□ =70%**	**Average localization time of two robots t(s)**	**Time saving (%)**	**Average Successful rate ε**	**Increasing of successful rate**
Symmetric	**Single robot**	368.02	70.6%	44%	32%
	Multi-robot	109.56		76%	
Asymmetric	**Single robot**	31.4	28.3%	86%	3%
	Multi-robot	22.485		89%	
Symmetric with obstacle	**Single robot**	33.24	27%	84%	4%
	Multi-robot	24.25		88%	
Asymmetric with obstacle	**Single robot**	28.74	31.6%	92%	1%
	Multi-robot	19.65		93%	

Table 4.6 shows the comparison results of Multi-robot localization with Single robot localization in simulated environment. The results demonstrate that our proposed approach applied in multi-robot localization benefit from exchanging information gathered by other robot other than itself. The improvements demonstrate two results: the time for localization and the successful rate for each robot. For example, with the help of

another robot, the time for localization is reduced by 70.6% in symmetric environment, by 28.3% in asymmetric environment, by 27% in symmetric environment with obstacle, and by 31.6% in asymmetric environment with obstacle. As to the successful rate, the largest increasing rate 32% shows in symmetric environment, and the other three are 3%, 4%, and 1% respectively.

(c) Study of parameter η

In this group of experiments, we focus on the parameter η. As we mentioned the value of η determines the stop point of each robot's localization which is specified by the minimum percentage of total particles contained in the largest cluster. It is not only used for terminating the localization process, but also used as a sign of to what extent the robot will believe it has been well localized. We apply our proposed approach in four simulated environments shown in Figure 4.7 under a different value of η, which is set to be 80%, to see how it can affect the performance.

The other parameters are the same with the previous experiments: dissimilarity threshold θ is set to be 60 pixels, threshold γ is set to 120 pixels, and the total number of particles is 5000. We repeat each experiment 50 times and compare the results with the previous cases while η is equal to 70%. Table 4.7 summarizes the comparison results.

Table 4.7: Comparison of multi-robot localization under two values of η (70% and 80%) using our proposed approach in four simulated environments.

Comparison of Multi-robot localization under two values of η						
Environment	**Multi robot**	**Average localization time of two robots t(s)**	**Time increasing (%)**	**Average Successful rate ε**	**Increasing of successful rate**	**Average error distance of two robots d(pixel)**
Symmetric	**η =70%**	109.56	18.1%	76%	2%	9.925
	η =80%	129.41		78%		9.025
Asymmetric	**η =70%**	22.485	14.5%	89%	3%	11.975
	η =80%	25.75		92%		10.06
Symmetric with obstacle	**η =70%**	24.25	16.9%	88%	4%	13.06
	η =80%	28.36		92%		11.38
Asymmetric with obstacle	**η =70%**	19.65	8.5%	93%	1%	14.695
	η =80%	21.33		94%		13.195

We can see from Table 4.7, compare to the experimental results under η is 70% in four environments, both average localization time and successful rate show different degrees of growth in the cases of η is 80%. The relatively larger increasing is shown in localization time, which is increased by 18.1% in symmetric environment, by 14.5% in asymmetric environment, by 16.9% in symmetric environment with obstacle, and by 8.5% in asymmetric environment with obstacle. But in contrast the increasing degree of successful rate is very small, which is only 2%, 3%, 4%, and 1%, respectively. The results show that a larger value of η will take more time to succeed because it requires more information about its external environment to let more particles fall in the largest cluster until it reach the stop point. In the mean time, the fact that more particles fall in the largest cluster indicates more likely the pose of representative of this cluster represents the true pose of the robot, in other words, it means higher successful rate or effectiveness. However, here comes a trade-off between efficiency and successful rate of robot's localization controlled by parameter η. If η is too big, it will take significantly longer time to localize itself while achieving a relatively small increasing of successful rate. On the other hand, if η is too small, despite the reduction of localization time, it will be

useless without guaranteeing the successful rate. Therefore, choosing an appropriate value of η depends on the specific demands of different applications. As to the values of average error distances of two robots, decreasing with the range from 0 to 2 pixels show in all cases. This is because along with the increasing of localization time, robot has more chance to perceive the external environment to update its pose estimate so that it will increase the accuracy of localization.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

Cooperative multi-robot localization utilizes the abilities of sensing and communicating one with another to estimate the poses of robots relative to a given map of the environment. In this thesis, we propose a clustering based MCL for cooperative multi-robot localization, in which each robot maintains its own clustering based MCL algorithm, and communicates with each other whenever it detects another robot. We develop a new information exchange mechanism, which makes use of the information extracted from the clustering component, to synchronize the beliefs of detected robots. This mechanism can deal with the delayed integration problem by avoiding unnecessary information exchange whenever detection occurs through the belief comparison. By doing so only those information exchanges that will benefit the localization process are allowed in our approach, to achieve improving the effectiveness and efficiency of multi-robot localization. In addition, unlike many other centralized cooperative localization approaches, the characteristic of without fusion center and the instant communication between two detected robots allow our proposed approach to be potentially scaled to large group of robots and to high speed of operation. Moreover, robots themselves are implicitly used as landmarks rather than only external landmarks to achieve the objective of benefitting the localization process.

Compared with single robot localization, experimental results performed in both real and simulated environments demonstrate that our proposed approach applied in

cooperative multi-robot localization can improve the performance, especially when they are localized in highly symmetric environments.

5.2 Future work

In future work, there are some limitations that deserve further improving.

Failure of MCL: Our proposed approach inherits some limitations of MCL. It can fail in indicating wrong location if the map is symmetrical, or no particles around the true location of robot known as deprivation problem. Our current approach cannot verify whether the outcome is the correct location of robot. In the future work, we will do more measurements either by adding more powerful and accurate sensors (laser range sensors and camera) or by more motion control to verify whether the outcome location is the true pose of robot.

Active localization: The path setting of robots in current approach is simply passive. Robot's navigation does not aim to facilitate the localization process. Thus, one future work could be focused on designing a more robust way of movement based on already gathered information to achieve speeding up the localization process.

Scalability of robots: Current approach can only address two robots' cooperative localization because of the limitation of identifying individual robots. The real robots used in our experiments only equip with virtual wall sensor for sensing the infrared signals. If more than two robots meet together, our approach could not distinguish which two robots are detected with each other. This problem can be solved by adding camera sensors and attaching different labels to each robot. Another important point for scalability is that the status variable for detecting landmarks of each robot mentioned in chapter 3, it should be change to array type variables which can indicate status of all

possible pair of robots in order to deal with the situation: if two robots have already exchanged their beliefs with each other, both of them still can exchange information with other robots before they detect any landmarks.

BIBLIOGRAPHY

[1] 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* (2003), 65-95.

[2] A, Howard, M, J Mataric, and G, S Sukhatme. Localization for mobile robot teams using maximum likelihood estimation. *In Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (Sept.30-Oct.5 2002), 434-439.

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

[4] B, Everitt, S, Landau, and M, Leese. *Cluster Analysis*. Arnold Publisher, 2001.

[5] D, Fox. Adapting the Sample Size in Particle Filters Through KLD-Sampling. *The International Journal of Robotics Research.*, vol.22, no.12 (2003), 985-1003.

[6] D, Wu, J, Chen, and Y, Wang. Bring Consciousness to Mobile Robot Being localized. *Proceedings of the 2009 IEEE International Conference on Robotics and Biomimetics* (741-746 2009).

[7] D, Koller and R, Fratkina. Using learning for approximation in stochastic processes. *In Proc. of the International Conference on Machine Learning (ICML)* (1998), 287-295.

[8] D, Fox, W, Burgard, H, Kruppa, and S, Thrun. A Probabilistic Approach to Collaborative Multi-robot Localization. *Autonomous Robots on Heterogeneous Multi-Robot System*, vol.8, no.3 (June 2000), 325-344.

[9] Esha, D Nerurkar, Stergios, I Roumeliotis, and Agostino, Martinelli. Distributed Maximum A Posteriori Estimation for Multi-robot Cooperative Localization. *IEEE International Conference on Robotics and Automation* (May 12-17 2009), 1402-1409.

[10] F, Dellaert, D, Fox, W, Burgard, and S, Thrun. Monte Carlo Localization for Mobile Robots. *IEEE International Conference on Robotics and Automation (ICRA)* (1999), 1322-1328.

[11] H, Hose and H, L Akin. The Reverse Monte Carlo localization algorithm. *Robotics and Autonomous Systems* (2007), 480-489.

[12] *http://www.irobot.com/sp.cfm?pageid=74*.

[13] J, Liu, K, Yuan, W, Zou, and Q, Yang. Monte Carlo Multi-Robot Localization Based on Grid Cells and Characteristic. *Proc of the 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, vol.8, no.3 (July 2005), 24-28.

[14] Maja, J Mataric. *The robotics primer*. The MIT Press, 2007.

[15] P, Trahanias and E, Scordalakis. An efficient sequential clustering method. *Pattern Recognition*, vol.22(4) (1989), 449-453.

[16] R, Luo, H, Min, M, Li, and Q, Huang. A Method for Active Global Localization in Multi-robot System. *International Journal of Advanced Robotic Systems*, vol.5, no.3 (2008).

[17] R, Kurazume, S, Nagata, and S, Hirose. Cooperative positioning with multiple robots. *In Proc. 1994 IEEE Int. Conf. Robotics and Automation*, vol.2 (May 8-13 1994), 1250-1257.

[18] S, I Roumeliotis. *Robust mobile robot localization: from single-robot uncertainties to multi-robot interdependencies*. Los Angeles,CA, 2000.

[19] S, Thrun, D, Fox, and W, Burgard. Monte carlo localization with mixture proposal distribution. *Proceedings of the AAAI National Conference on Artifical Intelligence* (2000), 859-865.

[20] S, I Roumeliotis and G, A Bekey. Distributed multi-robot localization. *IEEE Transactions on Robotics and Automation*, vol.18, no.5 (October 2002), 781-795.

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

[22] S, Thrun, W, Burgard, and D, Fox. *Probabilistic Robotics*. MIT Press, 2006.

[23] T, E Kurt. *Hacking Roomba*. WILEY Press, 2006.

[24] W, Burgard, D, Fox, D, Hennig, and T, Schmidt. Estimation the absolute position of a mobile robot using position probability grids. *Proc. AAAI-96* (1996).

[25] 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, ICARCV 06, 9th International Conference* , vol. 5-8 (December 2006), 1-7.

Appendix A: Results of Experiments

Raw Data of Experiments using iRobot Create

Single robot localization in symmetric environment using Create (□= 70%)			
	Localization Time(s)	**Error Localization**	**Error Distance(cm)**
1	283	fail	591
2	72		15
3	349	fail	320
4	458	fail	0
5	89	fail	533
6	245		10
7	287	fail	521
8	353	fail	0
9	155		15
10	77		19
11	96	fail	5
12	254		10
13	249	fail	9
14	304	fail	0
15	86		15
16	339	fail	145
17	154		14
18	355	fail	0
19	432	fail	0
20	213		18
21	122		10
22	88		16
23	430	fail	0
24	394	fail	0
25	106		9
26	128		5
27	302	fail	142
28	267	fail	194
29	96		11
30	91		15
31	230	fail	213
32	402	fail	0
33	125		13
34	443	fail	0
35	235	fail	565
36	223	fail	604
37	91		16
38	262	fail	512
39	255	fail	578
40	204		7
41	167		13
42	321	fail	358
43	94		19
44	213	fail	287
45	446	fail	0
46	118		12
47	233	fail	621
48	215	fail	345
49	148		8
50	245	fail	532
	Average time *t*	**Successful Rate □**	**Average Error Distance *d***
	230.88	42%	12.86

Single robot localization in asymmetric environment using iRobot Create (= 70%)			
	Localization Time (s)	**Error Localization**	**Error Distance (cm)**
1	50		18
2	60		15
3	82	fail	326
4	57		10
5	61		21
6	98	fail	490
7	82		9
8	122		23
9	59		31
10	102	fail	237
11	89		21
12	67		13
13	92		4
14	87	fail	523
15	81		14
16	58		11
17	90		21
18	86		6
19	79		12
20	105		5
21	93		21
22	59		16
23	87		16
24	110	fail	569
25	54		15
26	88		6
27	92		13
28	121		11
29	76		4
30	87		21
31	81		15
32	92		13
33	59		14
34	63		13
35	75	fail	321
36	79		18
37	81	fail	238
38	92		14
39	78		13
40	82		21
41	92		7
42	71		12
43	112		21
44	96		14
45	82		8
46	66	fail	420
47	53		16
48	81		12
49	79		13
50	92		4
	Average time *t*	**Successful Rate**	**Average error distance *d***
	81.6	84%	13.93

Single robot localization in symmetric environment with obstacle using iRobot Create (= 70%)			
	Localization Time(s)	**Error Localization**	**Error Distance(cm)**
1	49		10
2	132		5
3	120		17
4	109	fail	521
5	71		22
6	49		9
7	63		8
8	63		13
9	77	fail	233
10	86		18
11	55	fail	134
12	154		21
13	57		10
14	133		15
15	120		13
16	125		5
17	65		3
18	130	fail	0
19	79		18
20	123	fail	281
21	64		22
22	59		18
23	125		14
24	133		19
25	79		3
26	90		4
27	128		13
28	112	fail	546
29	131		8
30	58		19
31	98		15
32	54	fail	522
33	94		14
34	69		6
35	107		25
36	122		13
37	79		10
38	96		18
39	67		22
40	71	fail	636
41	102		14
42	72		17
43	66		8
44	59		8
45	162		18
46	54	fail	245
47	69	fail	573
48	156		21
49	61		5
50	134		15
	Average time *t*	**Successful Rate**	**Average Error Distance *d***
	92.62	80%	13.4

Single robot localization in asymmetric environment using iRobot Create (□= 70%)			
	Localization Time(s)	**Error Localization**	**Error Distance(cm)**
1	71	fail	327
2	64		13
3	60		5
4	47		20
5	89		14
6	57		15
7	101		21
8	67		14
9	82		9
10	66	fail	120
11	69		20
12	72		15
13	94		13
14	121		23
15	50		9
16	67		21
17	55		28
18	72		13
19	83		10
20	82		17
21	59		5
22	71		13
23	103	fail	423
24	58		6
25	63		14
26	82		19
27	78		14
28	72		15
29	49		10
30	59		21
31	64		14
32	89		20
33	62	fail	233
34	89		13
35	73		11
36	77		4
37	53		14
38	69		18
39	82		14
40	83		8
41	79		11
42	56		13
43	77		10
44	84		18
45	113		13
46	102	fail	539
47	89		14
48	58		11
49	82		20
50	75		12
	Average time *t*	**Successful Rate □**	**Average error distance *d***
	74.38	90%	14.11

Multi-robot localization in symmetric environment using iRobot Create (= 70%)						
	Localization Time(s)		Error Localization		Error Distance(cm)	
	Robot_A	Robot_B	Robot_A	Robot_B	Robot_A	Robot_B
1	90	81			11	5
2	164	158	fail	fail	589	612
3	160	187		fail	5	567
4	143	131	fail		294	19
5	159	134			12	13
6	121	152			4	8
7	95	112		fail	10	150
8	168	174			14	13
9	192	180	fail		298	14
10	165	177			19	20
11	142	165			4	2
12	86	89	fail	fail	435	523
13	143	149		fail	6	120
14	135	130			11	14
15	145	153			14	15
16	183	173	fail		239	13
17	170	166	fail		154	14
18	165	159			14	19
19	155	159			16	7
20	110	108			22	16
21	129	121	fail	fail	467	502
22	188	170		fail	13	533
23	97	102			16	14
24	132	144	fail		432	2
25	164	165			4	23
26	130	140		fail	11	527
27	143	154			21	7
28	176	160	fail		238	14
29	159	172			12	14
30	87	108	fail	fail	98	344
31	140	144			10	18
32	132	140			19	5
33	170	155	fail		278	11
34	187	166			10	12
35	150	143		fail	7	612
36	132	136		fail	21	520
37	135	148			17	19
38	150	140	fail		603	14
39	137	128			17	10
40	87	96		fail	19	554
41	148	145			20	14
42	133	140			9	12
43	159	166			15	17
44	160	157	fail	fail	282	367
45	112	105	fail		602	21
46	154	142			14	18
47	150	179			21	20
48	143	149			6	8
49	132	139	fail		624	14
50	163	143			17	18
	Average time *t*		Successful Rate		Average error distance *d*	
	143.4	144.68	70%	74%	13.17	13.43

Multi-robot localization in asymmetric environment using iRobot Create (= 70%)						
	Localization Time(s)		Error Localization		Error Distance(cm)	
	Robot_A	Robot_B	Robot_A	Robot_B	Robot_A	Robot_B
1	55	62			15	12
2	36	41			14	14
3	63	70	fail		358	5
4	64	68			9	17
5	67	69			18	9
6	71	75		fail	22	248
7	60	68			21	12
8	63	60			14	22
9	58	54			16	10
10	50	53	fail	fail	134	98
11	48	49			8	14
12	69	72			11	19
13	52	59			18	21
14	71	66	fail		636	9
15	62	72			15	12
16	59	62			16	20
17	72	61		fail	15	389
18	59	52			19	14
19	45	50			12	15
20	64	70		fail	20	213
21	71	75			14	14
22	70	72			2	9
23	45	40			9	20
24	55	61			15	14
25	66	69			21	10
26	60	64			14	19
27	44	39			15	14
28	63	68			16	19
29	56	48		fail	18	528
30	71	66			18	14
31	39	44	fail		442	18
32	49	54			14	16
33	60	65			0	19
34	68	59			14	15
35	70	62			15	6
36	70	64			17	15
37	75	79			8	12
38	72	77			15	19
39	68	72		fail	18	338
40	64	67			20	15
41	65	69			14	18
42	69	75	fail	fail	538	182
43	41	45			15	20
44	64	68	fail		601	15
45	59	63			14	0
46	35	40			9	12
47	68	69			12	21
48	66	62			22	14
49	56	58	fail		342	20
50	49	50			14	16
	Average time *t*		**Successful Rate**		**Average error distance *d***	
	59.92	61.54	86%	86%	14.56	14.63

Multi-robot localization in symmetric environment with obstacle using iRobot Create (= 70%)						
	Localization Time(s)		Error Localization		Error Distance(cm)	
	Robot_A	Robot_B	Robot_A	Robot_B	Robot_A	Robot_B
1	65	71	fail		204	5
2	87	82			13	8
3	72	74			13	15
4	64	60		fail	6	642
5	53	50			15	20
6	72	79	fail		541	17
7	71	74			14	19
8	69	74			10	22
9	84	79	fail		622	20
10	59	63			20	14
11	66	67			18	18
12	72	79			17	10
13	50	55		fail	5	539
14	72	77			18	8
15	80	84			13	19
16	82	76			9	24
17	78	72		fail	17	247
18	56	61			10	12
19	62	65	fail		389	5
20	74	82		fail	21	128
21	81	90			11	14
22	88	95	fail		332	14
23	78	67			18	15
24	65	60			14	17
25	58	50			12	19
26	57	59			8	20
27	62	66			10	14
28	62	57			18	11
29	59	64		fail	19	520
30	72	78			20	10
31	72	66			5	12
32	71	74			11	14
33	58	62			18	18
34	66	61	fail	fail	602	437
35	52	64			14	20
36	55	59	fail		255	14
37	88	90			15	17
38	81	87			18	19
39	75	79			17	10
40	62	70			10	11
41	77	61			8	14
42	72	80			15	7
43	80	72		fail	11	223
44	77	79			14	18
45	58	61			15	14
46	62	66			20	17
47	77	71			18	14
48	54	64		fail	18	189
49	65	71	fail		472	10
50	66	69			19	12
	Average time *t*		**Successful Rate**		**Average error distance *d***	
	68.76	70.32	84%	84%	14.17	14.55

Multi-robot localization in asymmetric environment with obstacle using iRobot Create (= 70%)						
	Localization Time(s)		Error Localization		Error Distance(cm)	
	Robot_A	Robot_B	Robot_A	Robot_B	Robot_A	Robot_B
1	45	50			10	15
2	65	62			14	14
3	52	54			8	19
4	44	48			15	19
5	34	31		fail	16	634
6	38	44			14	14
7	72	66			10	13
8	52	58			11	20
9	63	70	fail		522	12
10	59	50			21	15
11	45	47			22	6
12	55	58			18	14
13	65	62	fail		165	10
14	50	52			15	13
15	62	66			14	14
16	68	59			19	15
17	58	52			9	5
18	52	45			11	22
19	42	44			22	10
20	54	62			19	21
21	51	60			15	15
22	66	71			24	14
23	59	48			8	9
24	50	47			0	18
25	38	30			14	15
26	37	39			22	14
27	42	44			15	0
28	53	57	fail	fail	225	436
29	39	44			9	22
30	51	56			16	19
31	52	48			14	14
32	51	54			10	9
33	38	42			11	18
34	66	60			21	24
35	33	44		fail	16	504
36	35	39			18	12
37	62	68	fail		552	16
38	63	67			14	10
39	55	58			15	21
40	42	50			8	20
41	55	50		fail	20	613
42	52	58			14	19
43	60	51			15	15
44	55	58			24	24
45	36	40			15	8
46	42	44	fail		160	28
47	44	49			18	24
48	64	62			14	15
49	45	51			24	5
50	47	49			19	12
	Average time *t*		**Successful Rate**		**Average error distance *d***	
	51.16	52.36	90%	92%	15.13	15.02

Raw Data of Simulated Experiments

Simulated single robot localization in symmetric environment (= 70%)			
	Localization Time(s)	**Error Localization**	**Error Distance(pixel)**
1	313	fail	431
2	34		5
3	189		3
4	204	fail	336
5	234	fail	532
6	49	fail	503
7	544	fail	443
8	251		19
9	44		7
10	455	fail	537
11	410	fail	581
12	337	fail	342
13	39		12
14	98		15
15	780		8
16	289	fail	387
17	302	fail	443
18	1089	fail	507
19	69		0
20	147	fail	287
21	684	fail	412
22	127		13
23	305	fail	219
24	438		5
25	209		9
26	749	fail	554
27	34	fail	523
28	57		7
29	330		2
30	1407	fail	306
31	95	fail	489
32	39		5
33	892		8
34	309	fail	307
35	49	fail	471
36	24	fail	533
37	532		20
38	890	fail	549
39	640	fail	521
40	120		7
41	447		11
42	632		16
43	439	fail	507
44	332		2
45	872	fail	551
46	45		18
47	1082	fail	512
48	33	fail	290
49	27		5
50	685	fail	529
	Average time *t*	**Successful Rate**	**Average Error Distance *d***
	368.02	44%	8.95

Simulated single robot localization in asymmetric environment (= 70%)			
	Localization Time(s)	**Error Localization**	**Error Distance(pixel)**
1	30		3
2	36		9
3	24		9
4	33		3
5	23		10
6	50		13
7	40		3
8	35		4
9	33	fail	0
10	32		0
11	28		21
12	43		3
13	35		5
14	16		20
15	31		15
16	29		4
17	25	fail	62
18	30		6
19	48		2
20	39		25
21	70		14
22	19		21
23	25		8
24	35	fail	95
25	24		1
26	37		12
27	29		5
28	41	fail	0
29	30		20
30	17		6
31	25	fail	126
32	22		11
33	36		9
34	17		13
35	20		4
36	26		7
37	14		15
38	21		15
39	20		20
40	26		8
41	16		20
42	29		5
43	41		10
44	56	fail	172
45	55		17
46	36		21
47	14	fail	70
48	47		11
49	36		3
50	26		9
	Average time *t*	**Successful Rate**	**Average Error Distance *d***
	31.4	86%	10.23

Simulated single robot localization in symmetric environment with obstacle (= 70%)			
	Localization Time(s)	**Error Localization**	**Error Distance(pixel)**
1	23		15
2	33		13
3	45		0
4	38		6
5	25		19
6	38		11
7	56		4
8	33		11
9	22		2
10	39		19
11	43		4
12	26	fail	94
13	41		13
14	22	fail	103
15	43		7
16	33		22
17	27		8
18	23		16
19	39		9
20	21	fail	156
21	43		14
22	19	fail	618
23	24		23
24	43		20
25	33		11
26	41		11
27	13	fail	603
28	29		19
29	30		11
30	43		6
31	36		25
32	23	fail	605
33	32		7
34	23		18
35	39		14
36	40		5
37	27		16
38	42		11
39	27		1
40	57		13
41	38		6
42	36		26
43	23	fail	440
44	34		15
45	36		22
46	42		14
47	29		3
48	15		9
49	38		17
50	37	fail	134
	Average time *t*	**Successful Rate**	**Average Error Distance *d***
	33.24	84%	12.29

Simulated single robot localization in asymmetric environment with obstacle (= 70%)			
	Localization Time(s)	**Error Localization**	**Error Distance(pixel)**
1	29		4
2	24		7
3	19		26
4	25	fail	102
5	29		3
6	37		4
7	35		10
8	34		13
9	62		11
10	19		17
11	26		13
12	37		7
13	29		19
14	31	fail	112
15	26		9
16	32		16
17	25		20
18	19		17
19	33		21
20	22		7
21	36		17
22	40		7
23	26	fail	541
24	36		14
25	32		23
26	26		5
27	40		13
28	31		17
29	32		9
30	27		16
31	17		18
32	30		8
33	33		21
34	29		20
35	18		3
36	26		14
37	22		6
38	24		5
39	19	fail	135
40	22		24
41	16		6
42	27		11
43	43		13
44	23		17
45	18		21
46	34		16
47	28		3
48	25		6
49	30		16
50	34		15
	Average time *t*	**Successful Rate**	**Average Error Distance *d***
	28.74	92%	12.78

Simulated multi-robot localization in symmetric environment (= 70%)						
	Localization Time(s)		Error Localization		Error Distance(pixel)	
	Robot_A	Robot_B	Robot_A	Robot_B	Robot_A	Robot_B
1	71	155			1	11
2	167	83			12	10
3	39	81			9	3
4	104	23		fail	5	351
5	65	34			10	3
6	34	36	fail		642	6
7	304	23	fail	fail	282	0
8	171	515			9	5
9	13	25	fail		600	4
10	14	59			23	9
11	496	24			14	19
12	42	100			4	23
13	68	17		fail	11	287
14	121	116	fail	fail	183	178
15	21	25			14	8
16	122	415			10	6
17	19	26	fail	fail	370	599
18	51	90		fail	3	640
19	205	182	fail		358	2
20	41	20			5	11
21	67	106			23	5
22	342	147			14	11
23	174	152	fail		365	6
24	116	329			11	9
25	57	50	fail		343	22
26	137	128			2	7
27	172	169	fail		329	4
28	267	183		fail	6	622
29	212	21	fail		637	18
30	35	35			6	14
31	58	80			14	22
32	203	179		fail	10	641
33	85	89		fail	4	135
34	132	125	fail		344	23
35	129	87			23	11
36	263	234			6	19
37	71	68			3	7
38	112	110	fail		320	4
39	62	161	fail		463	8
40	125	136			3	12
41	106	178			9	14
42	19	17			5	5
43	39	30			3	4
44	81	90			23	2
45	108	88			3	23
46	54	50			23	4
47	81	88			21	6
48	151	20			8	11
49	21	10		fail	5	0
50	41	59		fail	9	192
	Average time t		Successful Rate		Average Error Distance d	
	113.76	105.36	74%	78%	9.84	10.03

Simulated multi-robot localization in asymmetric environment (= 70%)						
	Localization Time(s)		Error Localization		Error Distance(pixel)	
	Robot_A	Robot_B	Robot_A	Robot_B	Robot_A	Robot_B
1	50	30			7	11
2	29	22			9	18
3	23	22			13	17
4	21	15			19	14
5	24	11			10	8
6	10	31	fail		531	20
7	31	33			12	14
8	22	29			23	11
9	13	27			12	22
10	30	25			10	18
11	25	26		fail	16	93
12	27	26			6	13
13	17	18			9	20
14	20	11		fail	17	0
15	16	17	fail		0	30
16	28	29			9	9
17	26	27			10	15
18	28	27			19	10
19	14	20			22	18
20	17	11			2	8
21	17	28			16	8
22	34	32			9	10
23	14	17			17	3
24	14	31			8	14
25	13	23		fail	10	101
26	26	22			5	4
27	28	27			13	9
28	16	27			21	7
29	22	13			6	14
30	14	16			14	5
31	25	16			12	19
32	23	27			0	1
33	26	31			11	8
34	20	25			23	4
35	26	23	fail		79	20
36	27	25			2	3
37	23	27			0	11
38	27	28			11	18
39	15	30		fail	10	245
40	26	11			21	9
41	26	18			13	14
42	27	27			6	11
43	8	16			16	5
44	22	27	fail		0	9
45	13	23		fail	19	0
46	28	28			9	3
47	16	23	fail	fail	98	125
48	26	24			11	21
49	25	12			7	13
50	26	13			18	14
	Average time *t*		**Successful Rate**		**Average Error Distance *d***	
	22.48	22.94	90%	88%	11.84	12.11

Simulated multi-robot localization in symmetric environment with obstacle (= 70%)						
	Localization Time(s)		Error Localization		Error Distance(pixel)	
	Robot_A	Robot_B	Robot_A	Robot_B	Robot_A	Robot_B
1	33	18			9	5
2	30	27			1	12
3	24	17	fail	fail	128	175
4	16	30			6	11
5	46	34			15	17
6	32	24	fail		201	18
7	14	14			8	24
8	48	22			16	10
9	27	18			20	1
10	33	29			4	6
11	16	30			12	21
12	29	35			19	18
13	21	38			2	11
14	25	27		fail	11	0
15	14	33			20	17
16	33	16			23	8
17	18	37			8	17
18	15	23			9	20
19	17	27			17	18
20	24	24			4	5
21	14	40			11	7
22	12	18			5	13
23	13	20		fail	9	621
24	42	18			18	1
25	17	34			29	16
26	13	22			14	18
27	11	13			12	4
28	39	29			7	25
29	22	23		fail	13	104
30	18	41			17	19
31	22	16			12	17
32	27	14	fail		0	21
33	36	43			4	3
34	13	17	fail		628	10
35	37	39			18	22
36	16	22			17	3
37	30	33			20	15
38	12	10			15	8
39	25	32			17	19
40	18	27	fail		184	3
41	28	13		fail	3	521
42	22	25			7	18
43	14	24			19	9
44	16	18			17	17
45	39	24			21	18
46	32	13			18	21
47	14	13	fail		507	10
48	21	24			11	12
49	34	15		fail	19	593
50	13	37			22	3
	Average time *t*		**Successful Rate**		**Average Error Distance *d***	
	23.7	24.8	88%	88%	13.16	12.98

Simulated multi-robot localization in asymmetric environment with obstacle (□= 70%)						
	Localization Time(s)		Error Localization		Error Distance(pixel)	
	Robot_A	Robot_B	Robot_A	Robot_B	Robot_A	Robot_B
1	23	23			5	12
2	23	26			12	17
3	25	22			5	24
4	19	28			14	14
5	15	20	fail	fail	307	434
6	24	16			27	25
7	19	29			6	17
8	21	23			15	14
9	17	11			14	9
10	15	20			12	14
11	29	12			13	24
12	19	22			34	6
13	18	26		fail	9	287
14	29	28			19	12
15	25	28	fail		0	21
16	19	18			9	6
17	23	29			21	15
18	27	16			18	20
19	24	23			5	12
20	11	24			22	4
21	20	13			12	12
22	20	17			24	25
23	9	10			21	8
24	19	21			22	16
25	21	8			4	8
26	27	28			6	2
27	29	29			13	12
28	16	9			8	18
29	13	22			21	9
30	12	19			23	26
31	24	27			14	13
32	22	26			17	6
33	22	23			22	18
34	11	16			14	23
35	24	22			11	10
36	21	21			8	16
37	13	9		fail	25	502
38	18	22			8	22
39	21	10			17	11
40	12	19			12	6
41	11	18	fail		0	18
42	21	21			25	27
43	12	23			14	7
44	22	22			21	24
45	15	11			6	22
46	10	12			14	16
47	24	20			20	6
48	13	18			13	5
49	25	20			4	24
50	22	11		fail	12	398
	Average time *t*		Successful Rate □		Average Error Distance *d*	
	19.48	19.82	94%	92%	14.70	14.70

Simulated multi-robot localization in symmetric environment (□= 80%)						
	Localization Time(s)		Error Localization		Error Distance(pixel)	
	Robot_A	Robot_B	Robot_A	Robot_B	Robot_A	Robot_B
1	96	17			1	5
2	34	29			8	6
3	14	41			24	21
4	69	151	fail		537	3
5	195	13		fail	3	0
6	177	159			6	14
7	44	84			22	9
8	76	55	fail		320	9
9	155	305			7	11
10	16	67			3	8
11	233	196			17	1
12	37	14		fail	13	0
13	34	53			5	20
14	387	86	fail		455	3
15	298	109	fail		557	2
16	276	267		fail	16	457
17	102	125			18	13
18	55	38			1	7
19	19	99			4	9
20	60	78		fail	2	456
21	150	157			7	4
22	37	360		fail	12	0
23	316	56		fail	11	644
24	21	53			6	2
25	144	125	fail	fail	595	393
26	36	190			4	7
27	166	130	fail		641	15
28	233	154			9	4
29	16	21			7	18
30	318	81			18	6
31	276	280	fail		597	6
32	35	40		fail	14	0
33	31	56			11	5
34	185	61			9	15
35	80	85			1	14
36	298	153		fail	7	0
37	66	59	fail		334	18
38	217	257			6	9
39	110	391		fail	2	632
40	14	21	fail		238	14
41	291	169			1	5
42	14	141			13	10
43	54	36			2	14
44	126	104			3	6
45	273	367			8	10
46	341	43	fail	fail	287	389
47	84	208		fail	9	241
48	109	60			4	22
49	85	122			16	12
50	316	156			12	4
	Average time t		Successful Rate □		Average Error Distance d	
	136.38	122.44	80%	76%	8.55	9.5

Simulated multi-robot localization in asymmetric environment (= 80%)						
	Localization Time(s)		Error Localization		Error Distance(pixel)	
	Robot_A	Robot_B	Robot_A	Robot_B	Robot_A	Robot_B
1	31	32			7	7
2	31	26			9	13
3	28	31			9	4
4	30	43			4	22
5	16	25			2	11
6	32	32			12	16
7	28	32	fail		98	7
8	28	13			12	2
9	30	20			3	20
10	54	22			8	3
11	27	16			10	8
12	18	32		fail	26	114
13	33	28			12	9
14	22	52			18	9
15	26	32			11	12
16	23	27	fail		0	20
17	27	24			7	12
18	29	27			9	11
19	27	27			8	16
20	29	31			12	5
21	19	20			21	17
22	18	19			11	18
23	26	25			5	13
24	24	9		fail	18	117
25	27	30			14	14
26	17	13	fail		98	21
27	27	33			21	5
28	18	28			11	8
29	25	31			12	5
30	25	26			8	12
31	28	29			11	2
32	13	29			23	12
33	12	25			1	3
34	29	32			9	9
35	31	28			10	4
36	28	20			14	22
37	27	26			8	13
38	26	14		fail	9	0
39	18	17			13	16
40	29	17			10	18
41	18	31			18	7
42	22	30	fail		0	10
43	33	31			6	4
44	21	31		fail	15	112
45	30	29			1	5
46	11	25			13	15
47	30	27			8	5
48	16	13			13	3
49	24	22			1	14
50	26	26			10	0
	Average time *t*		Successful Rate		Average Error Distance *d*	
	25.34	26.16	92%	92%	10.72	10.48

Simulated multi-robot localization in symmetric environment with obstacle (= 80%)						
	Localization Time(s)		Error Localization		Error Distance(pixel)	
	Robot_A	Robot_B	Robot_A	Robot_B	Robot_A	Robot_B
1	22	15			6	4
2	39	45			12	19
3	33	20			8	6
4	41	16			5	16
5	16	15			20	7
6	41	20			13	11
7	20	27			9	19
8	28	30			16	2
9	18	21	fail		318	9
10	28	20			3	15
11	12	17			8	3
12	26	20			16	12
13	14	19			4	20
14	16	21			16	7
15	23	34			11	6
16	30	27			6	17
17	46	50			14	1
18	19	49			6	10
19	34	22		fail	14	497
20	41	52	fail		340	11
21	17	33			17	20
22	18	19			3	0
23	35	13			20	7
24	31	21			6	13
25	19	20			3	21
26	21	25	fail		282	12
27	20	12		fail	17	300
28	30	11			1	3
29	44	42			12	14
30	34	37			20	8
31	17	33			2	4
32	48	39			19	12
33	35	45		fail	20	304
34	47	43			7	8
35	25	33			23	17
36	33	31	fail		305	20
37	41	37			16	2
38	32	25			1	15
39	23	56			17	21
40	22	25			3	5
41	28	25			22	18
42	26	30		fail	17	467
43	23	43			4	21
44	15	16			23	17
45	36	18			11	10
46	62	53			4	12
47	25	28			17	4
48	21	19			9	11
49	18	25			15	6
50	22	24			14	21
	Average time *t*		Successful Rate		Average Error Distance *d*	
	28.3	28.42	92%	92%	11.52	11.24

Simulated multi-robot localization in asymmetric environment with obstacle (= 80%)						
	Localization Time(s)		Error Localization		Error Distance(pixel)	
	Robot_A	Robot_B	Robot_A	Robot_B	Robot_A	Robot_B
1	16	25			15	17
2	10	23			17	3
3	22	21			6	18
4	13	33			5	10
5	13	40		fail	24	405
6	11	13			18	8
7	34	22	fail		357	8
8	21	25			10	22
9	15	31			15	19
10	12	21			2	18
11	15	20			5	22
12	22	18			23	13
13	23	21		fail	18	531
14	23	17			9	20
15	14	22			26	10
16	20	18			15	20
17	17	15			3	8
18	19	21			28	6
19	21	23			5	21
20	24	16			22	5
21	17	21			14	6
22	33	28	fail		482	21
23	19	17			23	27
24	21	21			13	17
25	24	13			2	18
26	25	22			13	11
27	14	24			1	9
28	16	22			22	5
29	13	18			16	14
30	22	24			10	17
31	24	17			7	4
32	21	28			19	23
33	16	21			22	10
34	16	18			14	17
35	22	15		fail	6	377
36	28	19			3	3
37	24	24			12	19
38	21	20			15	2
39	31	29			22	11
40	35	34			17	22
41	28	24			14	19
42	19	23			8	14
43	26	19			12	3
44	16	28			14	18
45	25	28			21	4
46	23	21			18	13
47	26	24			1	22
48	28	19	fail		407	5
49	16	18			3	13
50	19	16			15	2
	Average time *t*		**Successful Rate**		**Average Error Distance *d***	
	20.66	22	94%	94%	13.26	13.13

VITA AUCTORIS

NAME: Guanghui Luo

PLACE OF BIRTH: Guangxi, China

YEAR OF BIRTH: 1985

EDUCATION: Northeastern University at Qinhuangdao, Hebei, China

2004-2008 Bachelor;

University of Windsor, Windsor, Ontario

2008-2011 Master.