2008

An Improved Approach For Multi-Robot Localization

Huaicheng Su
University of Windsor

An Improved Approach For Multi-Robot Localization

by

Huaicheng Su

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

2008

Your file Votre référence
ISBN: 978-0-494-47036-7
Our file Notre référence
ISBN: 978-0-494-47036-7

Author's Declaration of Originality

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

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

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

Abstract

Cooperative multi-robot localization techniques use sensor measurements to estimate poses (locations, orientations) of robots relative to a given map of the environment. Existing approaches update a robot's pose instantly whenever it detects another robot. However, such instant update may not be always necessary and effective, since both robots' pose estimates could be highly uncertain at the time of the detection. In this thesis, we develop a new information exchange mechanism to collaborative multi-robot localization. We also propose a new scheme to calculate how much information is contained in a robot's belief by using entropy. Instead of updating beliefs whenever detection occurs, our approach first compares the beliefs of the robots which are involved in the detection, and then decide whether the information exchange is necessary. Therefore, it avoids unnecessary information exchange whenever one robot perceives another robot. On the other hand, this approach does allow information exchange between detecting robots and such information exchange always contributes positively to the localization process, hence, improving the effectiveness and efficiency of multi-robot localization. The technique has been implemented and tested using two mobile robots as well as simulations. The results indicate significant improvements in localization speed and accuracy when compared to the single mobile robot localization.

Keywords: multi-robot, localization, Monte Carlo, belief, entropy, density estimation

Dedication

This thesis is dedicated to my parents who have supported me all the way since the beginning of my studies.

This thesis is specially dedicated to the people who lost their lives in SICHUAN earthquake, May, 2008.

Also, this thesis is dedicated to my fiancée Yihan Wang who has been a great source of motivation and inspiration.

Finally, this thesis is dedicated to my lovely cats, milk and simba.

Acknowledgements

First of all, I would like to express my deep and sincere gratitude to my supervisor, Dr. Dan Wu, for his invaluable guidance and advices, for his enthusiastic encouragement and his great patience to me. Without his help, the work presented here would not have been possible.

Secondly, I would also like to thank my external reader, Dr. Jonathan Wu, my internal reader, Dr. Alioune Ngom, and my thesis committee chair, Dr. Jessica Chen for spending their precious time to review this thesis and putting down their comments, suggestions on the thesis work.

Thirdly, I want to acknowledge my friends. Their wisdom guidance helped me solve many difficult problems during my research.

Finally, I am very grateful for my parents, Wei Wang and Luqun Su. During these two years' study, they gave me great confidence and solid support whenever I met any kind of problems. They always stand by my side and back me up no matter what happens. This is the most valuable asset in my life. Especially, I would like to give my special thanks to my fiancée Yihan Wang whose patient love enabled me to complete this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1
Introduction

Most mobile robot tasks require a robot to have accurate information about its localization through its sensors. Therefore, sensor-based mobile robot localization has been recognized as one of the fundamental problems in the research of mobile robotics [13]. More formally, mobile robot localization is the problem of estimating a robot's pose (location, orientation) in a global coordinate system, given a map of its environment and the history of its sensory and odometry readings [14]. The mobile robot localization problem comes in many different flavors [3, 15]. The most simple localization problem – which has received by far the most attention in the literature – is position tracking [3, 5, 81]. Here the initial pose of the robot is known, and the problem is to compensate incremental errors in a robot's odometry. More challenging is the global localization problem [5, 22, 41], where a robot is not given its initial pose but instead has to determine it from scratch. Significant efforts have been put to solve the global localization problem and a variety of effective techniques have been developed [6, 10, 23, 41]. Thereinafter, the proposed method in this thesis will focus on global localization.

Kalman Filters (KF) [19, 24, 33, 52], Extended Kalman Filters (EKF) [8, 22, 40, 41], Markov localization [6, 15, 21, 23, 26, 34, 36] and Monte Carlo Localization (MCL) [10, 57] are all well known approaches to solve the localization problem. In position tracking where the uncertainty of the position of a robot can be modeled using a unimodal distribution, Kalman and Extended Kalman Filters can be used effectively for localization. However, in the problem of global localization, using a unimodal distribution will fail. Monte Carlo localization has proven to be effective in dealing with problems where multimodal distributions are required to model the position of the robot [10]. A multi-robot system has some apparent advantages over a single robot system. Fox example, a multi-robot system can collect and integrate

multiple sensory information from different robots in the system.

Most of the existing works virtually address localization of a single robot only. The problem of cooperative multi-robot localization is not fully explored. At first glance, one could solve the problem of localizing N robots by localizing each robot independently [13]. Nonetheless, if robots can detect each other, there is an opportunity to do better. When a robot detects the location of another robot relative to itself, both robots can refine their internal beliefs based on the other robot's estimate by exchanging information, hence improve their localization accuracy respectively. The ability to exchange information during localization is particularly attractive within the context of global localization, where the detection of one robot can possibly reduce the uncertainty of the other robot in the estimated location dramatically.

To solve the multi-robot localization problem, researchers have proposed some methods. Amongst them, a probabilistic approach to collaborative multi-robot localization [13] is one of the most important ones, because the probabilistic nature of this approach makes it possible that teams of robots perform global localization in a real time fashion [13]. This method is based on Markov localization, a family of probabilistic approaches that have been applied with great success to single robot localization [4, 15, 28, 59]. Moreover, their approach uses sampling based representation [2, 10], which is able to approximate complex, multi-modal belief representations in real time. The authors also emphasize that information exchange between robots is important for cooperative multi-robot localization. To transfer information across different robots, probabilistic "detection model" is used to model the robots' abilities to recognize each other. When one robot detects another, these detection models are used to synchronize the individual robot's belief, thereby reducing the uncertainty of both robots during localization.

The above probabilistic method of cooperative multi-robot localization updates a robot's pose immediately whenever it perceives another robot. However, this instant

update may not contribute positively to the localization process. For example, if one robot detects another one and both robots' internal beliefs are highly uncertain, it might be more appropriate to delay the update or information exchange [13]. If later on one of the robots becomes much more certain and it detects another robot, then updating the belief of the detected robot could possibly speed up the localization of the detected robot.

In the following, the motivation of the thesis is first presented, after which the contributions of this thesis are highlighted, and the structure of the remaining chapters are outlined.

1.1 Motivation

Nearly all tasks of mobile robots require a robot to have accurate knowledge about its location. In order for a mobile robot to autonomously navigate, it must be able to localize itself. In other words, knowledge of position and orientation in the context of its surrounding is necessary for avoiding obstacles and developing path plans. Moreover, without knowledge of position, a mobile robot can not accurately execute its commands.

Recently, it has been noted that research efforts have been shifted from single robot to multi-robot systems. Multi-robot systems have been proposed for a variety of applications including space exploration, search and rescue, military surveillance, and hazardous cleanup [63]. One of the first problems that one needs to tackle in a multi-robot system is to localize each robot in the system. A multi-robot system has some obvious advantages over a single robot system. For instance, a multi-robot system can collect and integrate multiple sensory information from different robots in the system [42]. By integrating these multiple sensory data (also called sensor fusion); the system can possibly obtain better localization performance. The benefits of sensor

fusion are three-folded. Firstly, multi-robot can share their sensor information, which will increase the robustness of the localization algorithm for each robot. Secondly, the robots can exchange their pose estimates with each other, and use their geometric relationship to derive more reference information for localization [53]. Thirdly, different robots can be equipped with different type of sensors, so that the whole system can achieve more comprehensive environment characterization.

Most of the current research addresses localization of a single robot only. The problem of cooperative multi-robot localization is not fully explored. However, there are still some research works on multi-robot localization. The most commonly used approaches include Kalman Filters [43] and portable landmarks [38]. Most of the other research works are based on these works. In 2002, a new approach to collaborative multi-robot localization is presented by Fox and Thrun [13]. Each robot in its system maintains a probability distribution modeling its own uncertainty. When a robot detects the location of another robot relative to itself, both robots can possibly refine their internal beliefs based on the other robot's estimate, hence improve their localization accuracy. Nevertheless, the authors point out several limitations of their approach at the end of the paper. One of the limitations is that their method updates a robot's pose immediately whenever it perceives another robot. Because of this limitation, two robots will exchange their pose estimates in any case as long as they detect each other. However, this instant update may not contribute positively to the localization process. For example, if one robot detects another one and both robots' internal beliefs are highly uncertain, it might be more appropriate to delay the update or information exchange. If later on one of the robots becomes much more certain and it detects another robot, then updating the belief of the detected robot could possibly speed up the localization of the detected robot. In other words, if both robots have blurry knowledge of their poses at the detection time, it is not necessary to exchange their internal beliefs. Therefore, this approach suffers from the problem of *delayed information exchange* (also called *delayed integration* in [13]). This presents challenge and opportunity to develop an improved multi-robot localization approach

that can yield better localization results than conventional, single robot localization.

1.2 Contributions

This thesis is concerned with the problem of cooperative multi-robot localization in small-scale, particularly indoor, environments. The principal contributions of this thesis are as follows:

The primary contribution of this thesis is that we develop a new information exchange mechanism for collaborative multi-robot localization. We also propose a new scheme to calculate how much information is contained in a robot's belief by using entropy. In our approach, we study the problem of how multiple robots first compare with their beliefs, and then decide whether the information exchange is necessary. Our approach therefore avoids unnecessary information exchange whenever one robot perceives another robot. On the other hand, this approach does allow information exchange between detecting robots which contributes positively to the localization process, hence, improving the effectiveness and efficiency of multi-robot localization. Experimental results, carried out in real and simulated environments, demonstrate that our approach can reduce the uncertainty in localization significantly, when compared to conventional, single robot localization at lower sensor costs and relatively small communication overhead.

The second important contribution of this thesis is to overcome the sensor limitations of the given robots for implementing our proposed approach to multi-robot localization. Our approach extends the early work on Monte Carlo localization for single mobile robot localization. The Monte Carlo localization algorithm has been already successfully implemented by using many of the real robot platforms which are equipped with very powerful sensors, such as laser range finder, sonar and upward-pointed camera. Under the help of these sensors, robot can localize itself

much easier with more accuracy. In our proposed approach, the low-cost vacuum cleaning robots iRobot Discovery [65] and iRobot Create [66] are used. Both are the third generation iRobot robots which support the programming interface available for research and education. Comparing with a robot which equips with different powerful sensors, Roomba is only equipped with bumper sensors and virtual wall sensors for detecting the external environment and detecting other robots within this environment. Under the limit sensor's help, the implementation of our proposed approach would be a challenge. Therefore, the successful implementation of our method provides a cost-effective solution to apply complex robotic algorithms to these inexpensive robot platforms.

The third contribution of this thesis is that we develop a software application to carry out the proposed approach. The experimental results obtained through this application indicate feasibility of our approach in real and in simulation robots.

1.3 Guide to the Thesis

This thesis is organized as follows.

Chapter 2: Background. This chapter provides an introduction to the subjects that the proposed method builds upon. After explaining the idea of probabilistic robotics and uncertainty, the method of single mobile robot localization will be given. The Monte Carlo localization is specifically emphasized, since they constitute the core of the proposed approach. The attention then moves to the discussion of multi-robot localization.

Chapter 3: The Approach. The proposed multi-robot localization method based on Monte Carlo localization is presented in detail in this chapter. First the definition of the problem is described, followed by detailed presentation of the proposed approach.

Chapter 4: Experiments. The detailed information of the implementation and the experimental results will be described. The results section is divided into two main parts: experiments on the real robots and experiments in simulation. Finally, these experimental results are used for comparing with experimental results of single robot localization and the evaluations are obtained.

Chapter 5: Conclusion. This final chapter brings conclusion of the thesis and presents a sketch of possible future work.

Chapter 2
Background Knowledge

This chapter provides the background knowledge on which the proposed method is based. After explaining the idea of probabilistic robotics, we will address the problem of single mobile robot localization. We then explain one of the most important probabilistic algorithms for single mobile robot localization, namely, the Monte Carlo localization (MCL) algorithm. The MCL algorithm is the core of the proposed approach. Finally, current multi-robot localization methods are reviewed.

2.1 Uncertainty in Robotics

Robotics is the science of perceiving and manipulating the physical world through computer controlled devices [60]. Examples of successful robotic systems include mobile platforms for space navigation, search and rescue, and cars that drive by themselves [60], and many more. Robotic systems in the physical world are able to perceive information in their environment through sensors and manipulate through physical forces. The idea of intelligent manipulating devices has an enormous potential to better our life. Would not it be great if all the cars were able to safety travel by themselves, making car accidents a conception of the past? Would not it be wonderful if robots would take care of all the high risk tasks instead of human being? To be intelligent, in real world robotic applications, robots have to be able to accommodate a number of uncertainties [25] which exist in the physical world.

There are several elements that contribute to a robot's uncertainty [60]. First of all, robot environments are highly unpredictable. Second of all, sensors are restricted in what they can perceive. The resolution and range of a sensor is subject to physical limitations. Third, robot motion which involves motors is unpredictable. Uncertainty comes from effects such as control noise and mechanical malfunction. Fourth,

uncertainty may be caused by the robot's software application. All the internal models of the world are approximate. Models are abstraction of the real world. Therefore, they can only partially model the processes of the robot and its environment. Model errors are a source of uncertainty in robotics. Finally, uncertainty is further created through approximations of robotic algorithm. Because robots are real time systems, many robotic algorithms are approximations for achieving timely response by sacrificing accuracy. As robotics is more and more popular nowadays, managing uncertainty has become one of the most important steps for designing robust real world robot systems. This raises the question as how to deal with uncertainty [55] in robotics. What type of internal world models should robots use? And how should robots make decisions even if they are uncertain about their external world?

2.2 Probabilistic Robotics

The probabilistic approach to robotics addresses above questions. Probabilistic robotics is a fairly new approach to robotics which deals with uncertainty in robot observation and action. The key idea in probabilistic robotics is to represent uncertainty probabilistically. In particular, world models in the probabilistic approach are conditional probability distributions, which describe the dependence of certain variables on others in probabilistic terms. A robot's internal knowledge is also represented by probability distributions, which are derived by integrating sensor measurements into the probabilistic world models given to the robot [55]. Hence, probabilistic robotics provides an appropriate mechanism for coping with uncertainty. As the result, they surpass alternative techniques in many real world applications [12, 58, 62].

Probabilistic robotics has already achieved a great success in the field of robotics. For example, the 2005 DARPA Grand Challenge [67] was a prize competition for driverless cars, sponsored by the Defense Advanced Research Project Agency

(DARPA) [68], the most prominent research organization of the United Stated Department of Defense. The Grand Challenges [69] was the first long distance competition for driverless cars in the world. The U.S. congress authorized DARPA to offer prize money ($1 million) for the first Grand Challenge to facilitate robotic development. The prize money had been increased to US $2 million for the second Grand Challenge. This competition required each team of players to create the fully autonomous ground vehicles capable of completing a substantial off-road course within a limited time. Stanley [61] shown in Figure 2.1 is such kind of autonomous vehicle created by Stanford University's Stanford Racing Team. It competed in, and won, the 2005 DARPA Grand Challenge, earning the Stanford Racing Team the US $2 million dollar prize, the largest prize money in robotic history. Here, many of the probabilistic robotic algorithms are used to build the controller of the Stanley.

Figure 2.1: Stanley, the winner of the 2005 DARPA Grand Challenge. [70]

2.2.1 State

In probabilistic robotics, the world of a robot, or environment, is a dynamical system which includes state [55]. In other words, environments are characterized by

state which can be defined as the collection of all aspects of the robot and its environment that can impact the future. Throughout this thesis, state will be denoted x. The state at time t will be denoted x_t. Typical state variables used in this thesis are: (a) the robot's pose includes its location and orientation relative to a global coordinate. For mobile robots exploring in planar environments, the pose is usually given by three variables including its two location coordinates in the plane and its heading direction; (b) the location and features of surrounding objects in the environment are also state variables. An object may be a table, a wall or a door. Features of such objects may be their color or texture. For the problem studied in this thesis, the location of objects will be static; (c) in robot manipulation, the pose includes variables for the configuration of the robot's actuators. The robot configuration is often referred to as kinematics state. There are many other state variables that may impact a robot's operation. The list of potential state variables is endless. In most cases of robotic problems, state changes over time. Time, in this thesis, will be discrete, that is, all events will take place at discrete time steps $t = 0, 1, 2....$ If a robot starts its operation at a distinct point in time, we will denote this time as $t = 0$.

In most of the robotic applications, determining what to do is easy if one knows certain quantities [60]. For instance, moving a mobile robot is simple if the accurate position of the robot and all close by obstacles are known. Unfortunately, these variables are not directly measurable. Instead, a robot has to rely on its sensors to gather such kind of information. However, sensors carry only partial information about these quantities, and their measurements are noisy. As the result, the robot needs to maintain an internal knowledge respect to the state of its environment. Therefore, Estimating state from sensor data is the core issue of probabilistic robotics. State estimation addresses the problem of estimating quantities from sensor data that are not directly observable, but that can be inferred [60]. Thus, state estimation aims to recovery state variables from the sensor data.

2.2.2 Robot Environment Interaction

The robot can also influence its environment through its actuators. Each control action affects both the environment state, and the robot's internal knowledge with respect to this state. There are two basic types of interactions between a robot and its environment [35] shown in Figure 2.2: The robot is capable of gathering information about the state through its sensors, and it is also able to influence the state of its environment through its sensors. The first type of interaction is the process by which the robot uses its sensors to obtain information about the state of its surrounding. For example, a robot may take a camera image, a range scan or even use the bumper to touch the obstacle to obtain information about the state of the environment. The result of such an interaction called a measurement or an observation. The second type of interaction can be defined as the control actions which change the state of the world. Examples of control actions include robot motion and the manipulation of objects. A robot might keep all past sensor measurements and control actions. We will call such a collection as the data. In accordance to the two types of environment interaction, the robot has two different data streams.

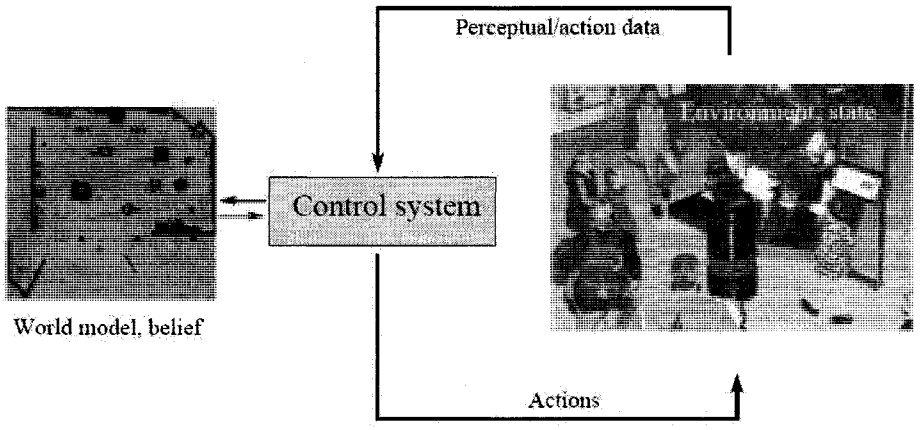


Figure 2.2: Robot environment interaction. [60]

The first stream is the measurement data which provides information about a

momentary of the environment. The measurement data at time t will be denoted as z_t. In this thesis, we assume the robot takes one measurement or observation at a time. The notation $z_{t_1:t_2} = z_{t_1}, z_{t_1+1}, z_{t_1+2}, \ldots, z_{t_2}$ denotes a collection of all measurements obtained from time t_1 to time t_2, where $t_1 \leq t_2$. The second stream is control data which carries information about the change of the state in the environment. One source of control data is odometer. Odometers are sensors that measure the revolution of a robot's wheel. Even though odometers are sensors, it is customary to treat it as the control data, because they measure the effect of the control action [60]. Control data will be denoted as u_t. As measurement data, we denote sequence of control data by $u_{t_1:t_2}$, where $t_1 \leq t_2$ $u_{t_1:t_2} = u_{t_1}, u_{t_1+1}, u_{t_1+2}, \ldots, u_{t_2}$. Both measurement data and control data play totally different roles in robotic system. Observation data provides information about the environment's state, thus it tends to increase the robot's knowledge. On the other hand, control data tends to bring a loss of knowledge due to the uncertainty in robot actuation and robot environment.

2.2.3 Belief

The above two different data streams can be used to estimate another important concept in probabilistic robotics, namely, belief. The belief reflects a robot's knowledge about the state of the environment. Probabilistic robotics represents beliefs using conditional probability distributions. Belief distributions are posterior probabilities over state variables conditioned on the data including measurement data and control data. We denote belief over the state variable x_t by $bel(x_t)$ which is a short form for the posterior $bel(x_t) = p(x_t \mid z_{1:t}, u_{1:t})$. This posterior is the probability distribution over the state x_t at time t, conditioned on all past measurements $z_{1:t}$ and all past controls $u_{1:t}$.

2.2.4 Probabilistic Laws

As described in the above section, the belief is used to represent a robot's knowledge about the state of its environment. Then the next question one may ask is how to evaluate the state? The evolution of state is controlled by probabilistic laws [60]. The state x_t is generated from the state x_{t-1}. At first glance, the state x_t may be conditioned on all past states, measurements, and controls. Therefore, the probabilistic law characterizing the evolution of state can be given by a probability distribution of the following form $p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t})$. An import fact is the following: If the state x is complete then it is a sufficient summary of all that happened in previous time steps [60]. In particular, x_{t-1} is a sufficient statistic of all previous measurements and controls up to this point in time, that is, $u_{1:t-1}$ and $z_{1:t-1}$. For all the variables in the expression above, only the control data u_t effects if we know the state x_{t-1}. In terms of probabilistic, this fact can be represented by the equation [60]

$$p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t}) = p(x_t \mid x_{t-1}, u_t). \quad (1)$$

The key point expressed by this equation is an example of conditional independence which states that the state variable x_t is independent of $u_{1:t-1}$ and $z_{1:t-1}$ if one knows the values of the previous state variable x_{t-1} and control data u_t, the conditioning variables. Based on the above description, one can also model the process by which how measurements are generated. If the state x_t is known, we have another important conditional independence equation [60]

$$p(z_t \mid x_{0:t}, z_{1:t-1}, u_{1:t}) = p(z_t \mid x_t). \quad (2)$$

This equation shows that the measurement z_t is only conditioned on the state variable x_t. Knowledge of any other variable such as previous measurements, controls, and previous states is not relevant.

We call $p(x_t \mid x_{t-1}, u_t)$ in Equation (1) as the *state transition probability* or *motion model* [60]. It specifies how state evolves over time with respect to robot control u_t and previous state x_{t-1}. The $p(z_t \mid x_t)$ in Equation (2) is called the *measurement probability* or *measurement model* [60]. It specifies the measurements z_t which are generated from the current state x_t. The state transition probability and the measurement probability together describe the complete system of the robot and its environment. Figure 2.3 illustrates the evolution of the state and measurements defined by these two conditional probabilities. The state at time t is dependent on the state at time $t-1$ and control u_t. The measurement z_t depends on the state at time t. Such kind of generative model is also known as dynamic Bayes network [18].

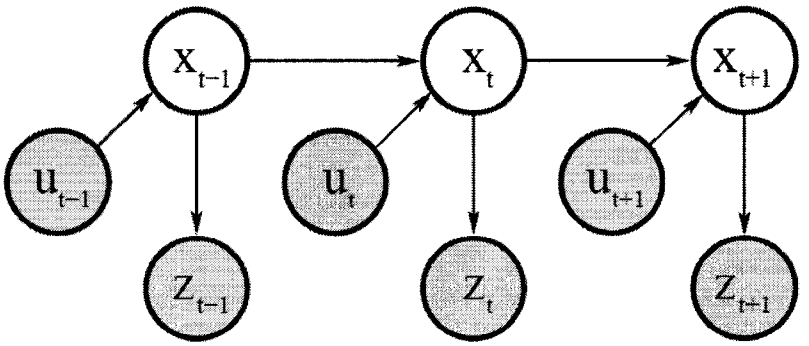


Figure 2.3: The dynamic Bayes network (DBN) which characterizes the evolution of states, controls, and measurements. [60]

2.2.5 The Bayes Filter Algorithm

The above two conditional probabilities: motion model and measurement model

are commonly used to estimate the belief in probabilistic robotics, and the basic algorithm for estimating beliefs in probabilistic robotics is using Bayes filter algorithm [60]. This algorithm calculates the belief distribution given measurement data and control data. Table 2.1 shows the basic algorithm of Bayes filter in pseudo code. From the table we know that the Bayes filter is recursive, by which the belief $bel(x_t)$ at time t is calculated from the belief $bel(x_{t-1})$ at time $t-1$. Its input is the belief at time $t-1$, along with the most recent control u_t and the most recent measurement z_t. Its output is the belief at time t. The Bayes filter algorithm includes two important steps. In line 3, it handles the control u_t. By doing so, it calculates a belief over the state x_{t-1} and the control u_t. One may notice that the equation in line 3 involving the state transition probability which transit the state from x_{t-1} to state x_t. And we call this update equation as the control update. The probability distribution $\overline{bel}(x_t)$ is often referred to as prediction in the context of probabilistic robotics [60]. It reflects the fact that $\overline{bel}(x_t)$ predicts the state at time t based on the previous state posterior, before incorporating the measurement at time t. The second important step of the Bayes filter is called the measurement update in line 4 in which the measurement probability is involved. It does so for each posterior x_t. By incorporating the state transit probability and measurement probability, one can calculate the final belief $bel(x_t)$ which is returned in line 6 of Bayes filter algorithm. To determine the posterior belief recursively, the Bayes filter algorithm requires an initial belief $bel(x_0)$ at time $t=0$. The initial belief characterizes the initial knowledge about the environmental state. In this thesis, we assign a uniform distribution to $bel(x_0)$.

In probabilistic robotics, Bayes filter algorithm is implemented in many different ways. There are quite a few algorithms are derived from the Bayes filter. Each one is

based on different assumptions of the measurement probability, the state transition probability, and the distribution of the belief [60]. In many robotic problems, beliefs have to be approximated because of real-time response requirement. Therefore, designing a suitable approximation algorithm is usually a challenging problem. When choosing an approximation, one has to trade off many different properties such as computational efficiency, approximation accuracy, and ease of implementation.

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

Table 2.1: The Bayes filter algorithm. [60]

2.3 Mobile Robot Localization

There are many existing implementations of Bayes filter. One of the important ones is the algorithm of mobile robot localization. Mobile robot localization is the problem of determining a robot's pose (localization, orientation) relative to the given map of the environment. The localization problem is a core problem in mobile robotics. It plays an important role in a variety of successful mobile robot systems [9, 17, 31, 39, 50]. Nearly all robotic tasks require knowledge about the location of the robot and the location of objects that are being manipulated. Thus, the mobile robot localization has been referred to as the most fundamental problem for providing a mobile robot with autonomous capabilities [7]. Figure 2.4 illustrates a graphical model for the single mobile robot localization problem. The robot is given a map of its

environment and its target is to determine its pose relative to this given map. In this Figure, the value of shaded nodes are known: the map m, the measurements z, and the controls u. The goal of localization is to infer the robot pose variable x.

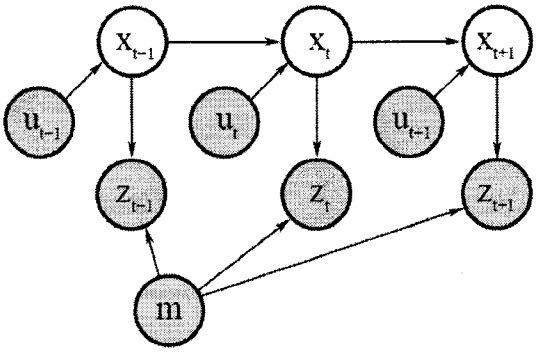


Figure 2.4: Graphical model of mobile robot localization. [60]

2.3.1 Classification of Localization Problems

The mobile robot localization problem comes in many different flavors. This classification divides localization problems along a number of important dimensions such as the property of the environment and the initial knowledge which a robot might have.

The first dimension to classify localization problems is by the type of knowledge that is available at the initial time to a robot. There are two cases under this dimension. The most simple localization problem is position tracking. Here the initial knowledge of robot's pose is known, and the problem is to compensate incremental errors in a robot's odometry. Algorithms of position tracking often rely on the assumption that the pose error is small. Within the context of position tracking, the pose uncertainty is often approximated by a uniform distribution like a Gaussian distribution. More challenging one is the global localization problem, where a robot is not given its initial pose but instead has to determine it by its own. The global localization problem is more difficult, since the error in the robot's estimate can not be assumed to be small.

As a result, a robot should be able to handle multiple, distinct hypotheses.

The second dimension is the environment which has a substantial impact on the difficulty of localization. Environments can be static or dynamic. Static environments are environments where the only variable quantity is the robot's pose. Only the robot moves in this environment while other objects remain at the same location at the same time. Dynamic environments possess objects other than the robot whose location or configuration changes over time [60]. Examples of such changes are: people, doors, and movable objects. Obviously, localization in dynamic environments is more difficult than localization in static ones. In this thesis, we focus on static environment.

The third dimension is whether or not the localization algorithm controls the motion of the robot. There are two cases. The first one is called *passive localization* [14], where the localization module only monitors the robot operating. The robot is controlled through some methods, and the robot's motion is not aimed at speeding up the process of localization. For instance, the robot may move randomly. The other one is called *active localization* [60] which controls the robot so as to minimize the localization error. This thesis entirely considers passive localization algorithm.

The fourth dimension is related to the number of robots in the problem. Single mobile robot localization is the most studied localization problem. It handles a single mobile robot only. Single robot localization collects all data at a single robot platform, and there is no communication involved. The multi-robot localization problem involves more than one robot. The research on multi-robot localization raises interesting problems such as representation of beliefs of multiple robots and the communication between a group of robots [13, 29, 43].

The above four dimensions describe the four important aspects of the mobile robot localization problem.

2.3.2 The map representation

The problem of mobile robot localization is based on a given map, which means that the map is initially known to the robot. In this section, we briefly explain the map representation within the context of mobile robot localization.

The problem of localization has been developed for a set of map representations. A map is a list objects in the environments along with its properties. Within the context of mobile robot localization, the most common used map representations are *feature-based map* and *localization-based map*. Feature-based maps can only specify the shape of the environment at the specific locations, that is to say the locations of the objects contained in the map. On the other side, location-based maps afford a label for any location in the environment. It keeps information not only about objects in the environment, but also about the absence of the objects like free space. In some problems, objects will be in the form of landmarks [13], which are distinct, stationary features of the environment that can be acknowledged reliably. Figure 2.5 shows the example of the location-based map and feature-based map respectively. In location based map (Figure 2.5(a)), the black areas are obstacles, and the white areas are free space. In feature based map (Figure 2.5(b)), the black dots mean landmarks, and the thin linkages indicate the topology of these landmarks.

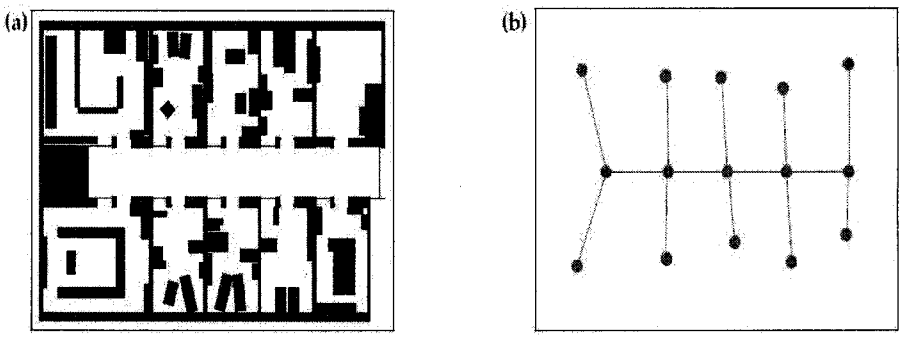


Figure 2.5: The map representation for robot localization: (a) A location-based map; (b) A feature-based map. [60]

2.3.3 Related Works

Because localization is a fundamental problem in the field of mobile robot, there are many existing probabilistic approaches to address the problem of single mobile robot localization. However, the majority of existing algorithms address only the position tracking problem. Among them, most of the earlier approaches employ Kalman filter [19, 24, 33, 52] based algorithms. These approaches are based on the assumption that the uncertainty in the robot's pose can be represented by a unimodal Gaussian distribution. They also exploit a range of restrictive assumptions such as Gaussian distributed noise and Gaussian distributed initial uncertainty. Under these assumptions Kalman filters provides extremely robust and efficient algorithm for the problem of position tracking. Nevertheless, the uncertainty in the robot's pose needs to be represented by multi-modal distribution in the global localization problem. Since the above Kalman filter algorithms can not represent multi-modal probability distributions, which makes inapplicable to global localization problem. This is one of the limitations for Kalman filter based algorithms.

This limitation is overcome by different approaches which have used increasingly richer schemes to represent uncertainty. These different approaches can be distinguished by the type of representation for the state space. Extended Kalman filters [8, 22, 40, 41] represent beliefs by using mixtures of Gaussians, thus enabling them to handle multiple, distinct hypotheses, each of which is represented by a separated Gaussian. Nevertheless, this approach inherits from Kalman filters which also exploit a range of restrictive assumptions such as Gaussian distributed noise. To meet this kind assumption, all practical implementations only extract low dimensional features from the sensor data, so discarding lots of the information acquired by the robot's sensor. Grid based Markov localization [15] can handle multi-modal and non Gaussian probability distribution at a fine resolution. Grid based methods perform numerical integration over an evenly spaced grid. These approaches represent beliefs

by piecewise constant functions like histograms over the space of all possible poses. Grid based methods are powerful, but suffer from excessive computation overhead and a priori commitment to the size and resolution of the state space. The computational requirements have an effect on accuracy as well, since not all measurements can be processed in real-time, and valuable information about the state space might be discarded.

It is noted that all the above algorithms share the same idea of probabilistic theory. They all estimate posterior distribution over the robot's poses under certain independence assumptions which will also be the case for the method illustrated in this thesis.

2.3.4 Monte Carlo Localization

Previous approaches were either computationally cumbersome, or had to resort to extremely coarse-grained resolutions. In this section, we review one of the latest and commonly used probabilistic approaches to single mobile robot localization called *Monte Carlo localization* (MCL). MCL constitutes the core of the proposed approach. MCL solves the global localization problem in an extremely effective and efficient way. Although it is relatively young, MCL has already become one of the most popular localization algorithms in robotics. It is easy to implement and works well across a broad range of localization problems.

The key idea of MCL is to represent the belief by a set of samples, drawn according to the posterior distribution over the robot's poses. That is to say, rather than approximating posteriors in parametric form such as Kalman filter, MCL represents the posterior by a collection of weighted samples which approximates the desired probability distribution. The idea of estimating state space recursively through samples is not new. In the statistical literature, it is known as particle filters [54].

Within the context of localization, the particle representation has a range of characteristics: (a) Particle filters can accommodate arbitrary sensor characteristics and noise distributions. (b) Particle filters are universal density approximations which weaken the restrictive assumptions on the shape of the posterior density when compared to previous parametric algorithms such as Kalman filters. (c) Particle filters focus computational resources in areas that are most relevant. (d) Particle filters control the number of samples online, which can adapt to available computational resources. (e) Finally, particles filters are easy to implement, which makes them an appealing pattern for mobile robot localization. The particle filter is a nonparametric implementation of the Bayes filter. The key idea of particle filter is to represent the posterior $bel(x_t)$ by a set of random state samples draw from this posterior. Such a representation is approximate, but it is nonparametric, and thus can represent a much broader space of distributions than Gaussian based algorithms. In particle filters, the samples of the posterior distribution are called particles and are denoted as $X_t = x_t^{[1]}, x_t^{[2]}, ..., x_t^{[M]}$. Each particle $x_t^{[m]}$ (with $1 \le m \le M$) is a concrete instance of the state at time t. M denotes the number of particles in the particle set X_t. In other words, a particle is a hypothesis as to what the true world state may be at time t.

1: **Algorithm Particle_filter**($\mathcal{X}_{t-1}, u_t, z_t$):
2: $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3: for $m = 1$ to M do
4: sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$
5: $w_t^{[m]} = p(z_t \mid x_t^{[m]})$
6: $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + (x_t^{[m]}, w_t^{[m]})$
7: endfor
8: for $m = 1$ to M do
9: draw i with probability $\propto w_t^{[i]}$
10: add $x_t^{[i]}$ to $\mathcal{X}_t$
11: endfor
12: return $\mathcal{X}_t$

Table 2.2: The particle filter algorithm. [60]

Because the intuition behind particle filters is to approximate the belief $bel(x_t)$ by the set of particles X_t, the probability for a state hypothesis x_t will be proportional to its posterior $bel(x_t)$ $x_t^{[m]} \sim p(x_t \mid z_{1:t}, u_{1:t})$ [60]. As the result, the more the number of samples falls in the region of the state space, the more likely it is the true state falls into this region. Because the particle filter is a recursive Bayes filter, it constructs the belief $bel(x_t)$ recursively from the previous belief $bel(x_{t-1})$. Since beliefs are represented by a set of particles, particle filters construct the particle set X_t recursively from the set X_{t-1}. Table 2.2 shows the particle filter algorithm. The input of this algorithm is the particle set X_{t-1}, along with the most recent control data u_t and the most recent measurement data z_t. Line 4 generates a hypothetical state $x_t^{[m]}$ for time t based on the particle $x_{t-1}^{[m]}$ and control u_t. This step involves sampling from the state transition distribution $p(x_t \mid x_{t-1}, u_t)$. Line 5 calculates the importance factor for each particle $x_t^{[m]}$. The importance factor [44] is non-negative numerical parameters for determining the weight (importance) of each sample, denoted $w_t^{[m]}$. Importance factors are used to incorporate the measurement z_t into the particle set. So the importance is the probability of the measurement z_t under the particle $x_t^{[m]}$. From line 8 to line 11, this algorithm implements what is known as resampling or importance sampling [51]. Resampling transforms a particle set of M particles into another particle set by incorporating the importance weights into the resampling process, the distribution of the particle change. The resampling step is a probabilistic implementation of the Darwinian idea of survival of the fittest [60]. It refocuses the particle set to regions in the state space with high posterior probability. By doing so, it focuses the computational resources to regions in the state space where most relevant.

Table 2.3 shows the basic MCL algorithm which is obtained by substituting the probabilistic motion model and measurement model into the particle filter algorithm. The basic MCL algorithm represents belief $bel(x_t)$ by a set of M particles $X_t = \{x_t^{[1]}, x_t^{[2]}, ...x_t^{[M]}\}$. The input of this algorithm is the particle set X_{t-1}, along with the most recent control data u_t, the most recent measurement data z_t, and the given map of the environment. The initial set of samples represents the initial belief $bel(x_0)$ about the state of the whole system. For instance, in global mobile robot localization, the initial belief is a set of poses drawn according to a uniform distribution over the robot's universe, annotated by the uniform importance factor $\frac{1}{M}$ to each particle. The line 4 samples from probability motion model by using particles from previous belief as a starting point. The probability measurement model is then applied in line 5 to determine the importance factor (weight) of that particle. The above sampling process is repeated m times, producing a set of m weighted samples $x_t^{[i]}(i = 1,2,...m)$ for current state space. Line 8 to line 11 is the resampling routine for MCL algorithm.

1:	**Algorithm MCL**$(\mathcal{X}_{t-1}, u_t, z_t, m)$:
2:	$\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3:	for $m = 1$ to M do
4:	$x_t^{[m]} = \textbf{sample\_motion\_model}(u_t, x_{t-1}^{[m]})$
5:	$w_t^{[m]} = \textbf{measurement\_model}(z_t, x_t^{[m]}, m)$
6:	$\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + (x_t^{[m]}, w_t^{[m]})$
7:	endfor
8:	for $m = 1$ to M do
9:	draw i with probability $\propto w_t^{[i]}$
10:	add $x_t^{[i]}$ to $\mathcal{X}_t$
11:	endfor
12:	return $\mathcal{X}_t$

Table 2.3: The algorithm of Monte Carlo localization. [60]

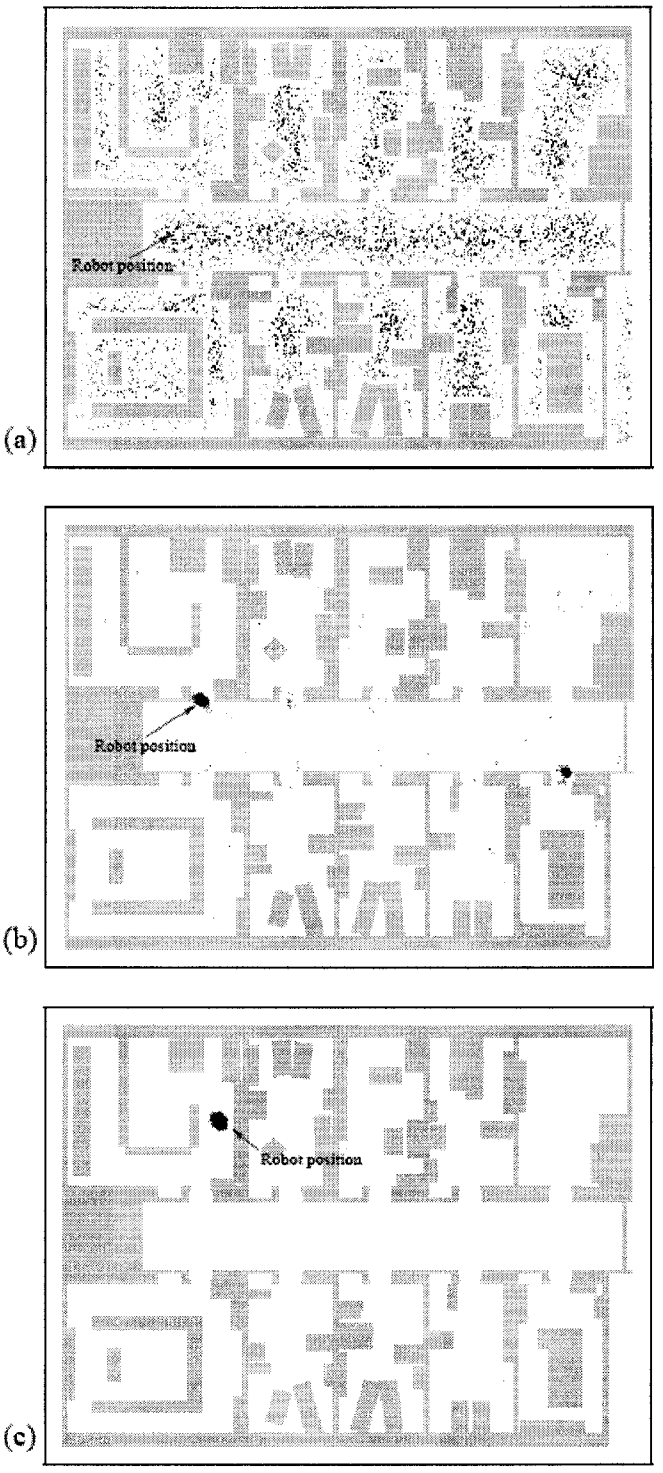


Figure 2.6: Global localization of a mobile robot using MCL: (a) Global uncertainty, particles are uniformly distributed; (b) Particles after approximately 1 meter of robot motion. Due to environment symmetry, most particles are centered on two locations; (c) After the robot enters one room, thus breaking the symmetry. [13]

Figure 2.6 shows an example of MCL within the context of global localization for single mobile robot in an office environment. This robot is equipped with laser range finders. It is also given a map of the environment. In Figure 2.6(a), the robot is initially globally uncertain; hence all the particles are spread uniformly through the free space. Figure 2.6(b) shows the particle set after approximately 1 meters of robot motion. Due to the symmetry of the environment, MCL has disambiguated the robot's pose centered on two locations. Finally, Figure 2.6(c) illustrates that after another 1 meter of robot motion, the robot has entered into one room. The ambiguity is resolved, and the robot knows where it is. The majority of samples are now centered closely on the correct position.

2.3.5 Multi-Robot Localization.

Recently, it has been noted that research efforts have been shifted from single robot to multi-robot systems. Multi-robot systems have been proposed for a variety of applications including space exploration, search and rescue, military surveillance, and hazardous cleanup [63]. One of the first problems that one needs to tackle in a multi-robot system is to localize each robot in the system. A multi-robot system has some obvious advantages over a single robot system. For instance, a multi-robot system can collect and integrate multiple sensory information from different robots in the system [42]. By integrating these multiple sensory data (also called sensor fusion); the system can possibly obtain better localization performance. The benefits of sensor fusion are three-folded. Firstly, multi-robot can share their sensor information, which will increase the robustness of the localization algorithm for each robot. Secondly, the robots can exchange their pose estimates with each other, and use their geometric relationship to derive more reference information for localization [53]. Thirdly, different robots can be equipped with different type of sensors, so that the whole system can achieve more comprehensive environment description.

There are some existing algorithms addressing the problem of multi-robot localization. The most commonly used approaches include Kalman filters [19, 24, 33, 52] and portable landmarks [29, 38]. Most of the early works focus on the question of how to reduce the odometry error using a cooperative team of robots. One popular approach to cooperative robot localization is to use the mover and observer strategy, where two groups of robot are involved, a stationary one and a moving one. The measurements are then used to correct the odometry error accumulated by the moving robots. In some cases, odometry data is discarded and localization relies only on observations by the stationary robots. Within this context, Kurazume [29] introduced the notion of regarding robots as the portable landmarks. A similar method is presented in [38]. The authors deal with the problem of exploration of unknown environment using two mobile robots. In order to reduce the odometry error, one robot is equipped with a camera tracking system that allow it to determines its relative position and orientation with respect to a second robot which carries a helix target pattern and acting as a portable landmark. Although the mover and observer approach has proven successful, all approaches have the following limitation: (a) only one robot or a team of robots is allowed to move at a certain time instant; (b) the two robots or teams of robots must maintain visual contact at all the time; (c) this kind of approach slows down the overall speed. However, all these approaches only seek to reduce the odometry error. None of them incorporates environmental feedback into the estimation, and consequently they are unable to localize robots relative to each other, or relative to their environments. Even if the initial localizations of all robots are known, they will be getting lost ultimately.

Another approach is to allow all the robots to move at the same time. As an example of the method, the extended Kalman filter (EKF) approach to robot localization has been applied to cooperative localization by Roumeliotis and Bekey [43]. They present an approach to multi-robot localization in which sensor data from a heterogeneous collection of robots are combined through a single Kalman filter to estimate the pose of each robot in the team. They also show that how this centralized

Kalman filter can be broken down into n separate Kalman filters, one for each robot to allow for distributed processing. Their distributed approach allows a robot to store sensor information when not in contact with the group and to incorporate it whenever encounters happen. The motion model of the robots ensures propagation of position estimate and associated uncertainty when the robot can not observe any other robot, as well as consistent data fusion in case of a relative measurement. Therefore, this approach enables the group of robots to move continuously without having to be stayed within visible range at all the time.

Fox et al. [13] propose a statistical method for collaborative multi-robot localization. This approach extends their earlier work on MCL single mobile robot localization. Their method uses a sample based version of Markov localization, capable of localizing mobile robots in the real time fashion. To avoid exponential complexity in the number of robots, a factorial representation is used where each robot in their system maintains a probability distribution describing its own pose. When a robot determines the location of another robot relative to its own, both robots can refine their internal beliefs based on the other robot's estimate, thus improving their localization accuracy. The ability to exchange information during localization is particularly attractive in the context of global localization. To transfer information across different robots, the probabilistic detection model was proposed to model the robot's ability to recognize each other. When one robot detects another, these detection models are used to refine the individual robot's belief, consequently possibly reducing the uncertainty of both robots during localization. In [13], a combination of camera images and laser range scans are used to determine another robot's relative location. The reliability of the detection process is modeled by learning a parametric detection model from data, using the maximum likelihood estimator. During localization, detections are used to introduce additional probabilistic constraints which tie one robot's belief to another robot's belief. To combine sample sets generated at different robots in which each robot's belief is represented by a separated sample set, their approach transforms detections into density trees [27] which approximate discrete

sample sets by piecewise constant density functions. These trees are then used to help to refine the importance factors (weight) of other robot's belief, thus reducing their uncertainty in response to the detection. As the result, the robots are able to localize themselves faster and maintain higher accuracy.

Chapter 3
An Improved Approach For Multi-Robot Localization

3.1 Limitation of Existing Method

As reviewed in chapter2, Fox et al. [13] proposed a statistical method to collaborative multi-robot localization using MCL. This approach extends their earlier work on MCL for the single mobile robot localization [14]. Their approach in [13] uses a sample based version of Markov localization, capable of localizing mobile robots in a real time fashion. When teams of robots localize themselves in the same environment, probabilistic methods are used to refine each robot's belief whenever one robot detects another. However, the authors point out several limitations of their method in [13]. One of the limitations is that robots update their belief estimates constantly whenever one robot perceives another robot. Because of this limitation, two robots will exchange their pose estimates in any case as long as they detect each other. However, this instant update may not contribute positively to the localization process. For example, if one robot detects another one and both robots' beliefs are highly uncertain, it might be more appropriate to delay the update or information exchange. If later on one of the robots becomes much more certain and it detects another robot, then updating the belief of the detected robot could possibly speed up the localization of the detected robot. In other words, if both robots have blurry knowledge of their poses at the detection time, it is not necessary to exchange their internal beliefs. Therefore, this approach suffers from the problem of *delayed information exchange* (also called *delayed integration* in [13]).

3.2 The Proposed Method

In this chapter, we propose an approach to address the delayed information exchange problem. It is our belief that robots do not have to exchange information whenever they detect each other. Such information exchange should only occur when this exchange will benefit the localization process. Therefore, in our method, when one robot detects another robot, we first compare their beliefs to see which robot is more certain about its location, and then, based on the result of the comparison, we decide whether information exchange is necessary. In our approach, we assume that there is only a remote chance that more than two robots detect each other simultaneously, which happens rarely and will be ignored when it happens. In order to solve the problem of delayed information exchange, Fox et al. in [13] suggested that the robots are required to keep track of their actions and measurements after detecting other robots and landmarks. Following this suggestion, in the proposed approach we use a status variable of landmark detection to separate all robots within a team into two situations:

(a) The first situation is that a robot has already detected a landmark in the environment (the status variable of landmark detection is set to true). In this situation, a robot's belief usually becomes more certain by gathering some information of its pose relative to the environment through landmark detection.

(b) The second situation is that a robot has not yet detected any of the landmarks in the environment (the status variable of landmark detection is set to false). In this situation, a robot's belief about its pose is normally highly uncertain.

Based on the above two situations, our method divides the events of robots' detections with other robots into three groups:

(a) In the first group, both robots detect each other before they observe any of the landmarks in the environment. In this group, both robots' beliefs about their environment and their poses relative to this environment are highly uncertain. Therefore, there is no need to exchange their beliefs at the detection time.

(b) In the second group, when two robots detect each other, one of the robots has already detected a landmark, while another robot has not yet detected any of the landmarks. In this group, one robot has already obtained some knowledge about its pose relative to its surroundings by perceiving a landmark early, while another robot's belief is still very much uncertain. Thus, two robots will exchange their beliefs at the time of detection, which means that the former robot is about to use its belief which contains more information to refine the latter robot's belief which contains less information. By doing so, the former robot helps the latter robot to accelerate its localization process.

(c) In the last group, both robots have already detected landmarks prior to their detection to each other. In this group, both robots have some knowledge about their poses relative to their environment, so that they require comparing their beliefs during the detection to see which robot is more certain about its location. In order to compare the beliefs between two robots, our approach first applies density estimation to extract probability density for each robot's belief, and then make use of these probability density values to calculate how much information is contained with each robot's belief. Subsequently, the belief of the robot which contains more information is used to refine the belief of the robot which contains less information.

Our approach therefore avoids unnecessary information exchange whenever one robot perceives another robot. On the other hand, this approach does allow information exchange between detecting robots which contributes positively to the localization process, hence, improving the effectiveness and efficiency of multi-robot localization. The complete approach in the form of pseudo code is summarized in

Table 3.1. In the following sections, the detailed explanations of the above proposed approach will be provided.

Initially, each robot in the team does the single robot MCL, and each robot maintains a particle set to represent its own internal belief about its pose.

When two robots detect each other:

(a) if both robots have not perceived any landmarks before,

there is no belief exchange between two robots;

(b) if one robot has already detected the landmark before, while another robot has not detected any of the landmarks yet,

the former robot will make use of its current belief to help the latter robot to refine its belief;

(c) if both robots have already detected landmark,

both robots will calculate the entropy of their beliefs and make comparison:

if one robot's entropy of its belief is less than the other robot,

the former robot will help the latter robot to refine its belief;

else if one robot's entropy of its belief is greater than the other robot,

the former robot's belief will be refined by the latter robot's belief;

else

there is no information exchange between two robots.

If both robots have already exchanged their beliefs, they can not exchange their beliefs again until the next time after one of the robots observes the landmark or both robots detect the landmarks.

Process of calculating entropy of a robot's belief

(a) calculate the probability density of a particle set through kernel density estimation;

(b) plug these probability density values into the formula of entropy.

(c) this process will return the entropy value of a robot's belief.

Table 3.1: The proposed approach for multi-robot localization.

3.2.1 Problem Statement

In this section, we specify the multi-robot localization problem which will be addressed in this thesis. First of all, we state the following assumptions:

(a) A group of M independent robots move in a two-dimensional space. Each robot maintains its own belief information that models only its own uncertainty [13].

(b) Each robot carries both interior sensing devices such as odometry reader and exterior sensing devices such as bumper. These sensors measure the self motion of the robot and perceive the external environment for localizing features such as landmarks.

(c) There is a remote chance that more than two robots detect each other simultaneously, which happens rarely and will be ignored when it happens. (Our experiment in the next chapter justifies that this is a practical assumption.)

(d) The last assumption is that our robots all have the same sensors.

In this thesis, we study the problem of determining a principled way to develop the information exchange mechanism during the interactions between members of a group of robots. The information exchange between the robots is only necessary and useful when two robots detect each other and satisfy some conditions. To formulate the problem in such a way will be allowing for information exchange satisfying with minimal communications requirement and contributing positively to the localization process.

3.2.2 Explanations of Proposed Method

In this section, we explain our proposed method in detail. Because the core idea

of multi-robot localization is to incorporate measurements taken at different robots, each robot in the group can benefit from information gathered by other robots. At first glance, we could solve the problem of localizing N robots by localizing each robot independently. In our approach, each robot maintains its own belief which models only its own uncertainty. In the absence of detections, each member of a group of robots performs MCL independently. Detections are used to provide additional information between the two robots involved, which will be leading to refine local estimates of each robot.

As mentioned in the section 3.1, in order to solve the problem of delayed information exchange, our approach lets the robots keep track of their actions and measurements after detecting other robots or landmarks. By doing so, our method separates all the robots within a team into two situations based on the robot's status of landmark detection:

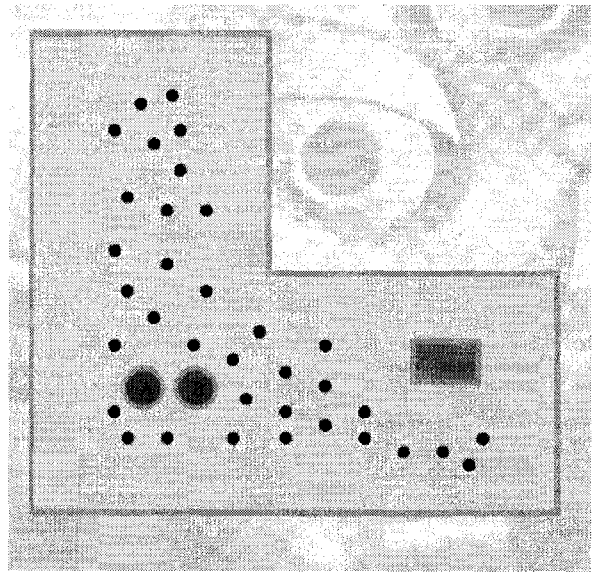
(a) The first situation is that a robot has already detected any of the landmarks in the environment. In our approach, we make use of a status variable of landmark detection to record whether a robot has detected a landmark or not. Initially, this status variable of landmark detection is set to false to reflect that a robot knows nothing about its environment. In this situation, our approach will set the status variable of landmark detection to true, which indicates that the robot becomes more certain about its pose relative to its environment by gathering some information of its environment through landmark detection.

(b) The second situation is that a robot has not yet detected any of the landmarks in the environment. In this situation, the status variable of landmark detection keeps its initial value, which means that the robot is still extremely uncertain about its pose relative to its environment. By using the status variable of landmark detection, our approach is capable of tracking the measurements after detecting the landmark.

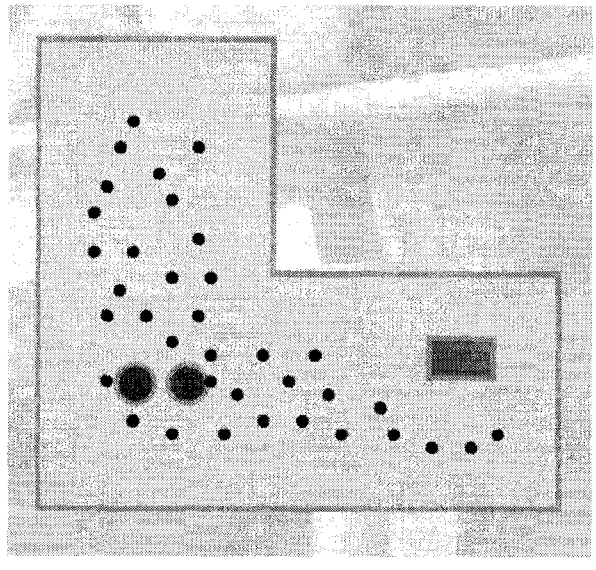
Based on the above discussion, our method divides the events of robots' detections to other robots into three scenarios:

(a) In the first scenario, both robots detect each other before perceiving any of the landmarks in the environment. Because the pair of robots' beliefs about their poses relative to their surroundings are both highly uncertain, it is not necessary to exchange their beliefs during the detection time. The information exchange will not make any positive contributions to the localization process of each robot at this time.

Figure 3.1 illustrates the first scenario where two robots detect each other without information exchange. In Figure 3.1, we have a 2D L-shaped environment which includes two robots *A*, *B* represented by two labeled dark circles (robot *A* and robot *B* are in the same environment) and a rectangle-shaped obstacle. Because our proposed approach is based on the MCL (the implementation details of motion model and measurement model based on our robot will be presented in the next chapter), the robot's belief is represented by a collection of weighted particles (the particles in Figure 3.1 are represented by a set of small black dot). For the sake of easy to demonstrate our three scenarios, the particle set in Figure 3.1(a) is used to represent the robot *A*'s current belief. Particle set in Figure 3.1(b) is used to represent the current belief of robot *B*. Initially, both robots are global uncertainty about their environment. In this scenario, although two robots wander around in the environment for a while, they both have not yet detected any of the landmarks, which indicates that both robots' beliefs are highly uncertain. When robot *A* and robot *B* detect each other within the context of scenario one, it is unnecessary to exchange any information between two robots. By doing so, we are able to save the computational resource from exchanging needless information compared with the method proposed in [13].



(a)



(b)

Figure 3.1: Two robots detect each other without information exchange: (a) The current belief of robot *A* is represented by a set of weighted particles (a set of small black dot); (b) The current belief of robot *B* is represented by a set of weighted particles. (Robot *A* and *B* are in the same environment)

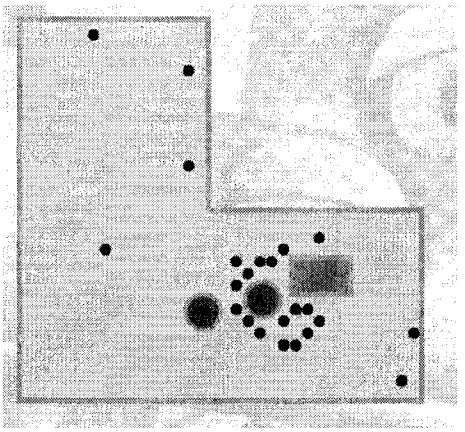
(b) In the second scenario during the time of two robots detecting each other, one of the robots has already detected a landmark, while another robot has not yet detected any of the landmarks. In this scenario, the former robot has already obtained some

knowledge about its pose relative to its environment by detecting a landmark in the early time, while the latter robot is still highly uncertain about its pose. Therefore, two robots will exchange their beliefs at the detection time. The former robot that has more certain belief will help the latter robot to refine its pose estimate.

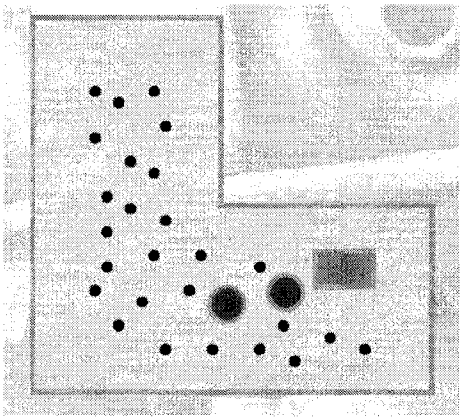
Figure 3.2 illustrates the second scenario where two robots detect each other with information exchange. In this Figure, we have exactly the same environment setup as in Figure 3.1. Figure 3.2(a) shows that robot A's belief becomes more certain by detecting a landmark in the environment. Because robot B has not yet detected any of the landmarks, the particle set in Figure 3.2(b) shows that the current belief of this robot is still highly uncertainty. Figure 3.2(c) illustrates the resulting particle set of robot B after exchanging information with robot A. This Figure demonstrates that the robot A makes use of its belief to help robot B to refine its belief. As the result, the localization process of robot B is accelerated.

(c) In the last scenario, both of the robots have already detected landmarks prior to their detection to each other. In this scenario, both robots have already obtained some knowledge about their poses relative to the environment. According to our approach described in section 3.2, we first compare these two robots' beliefs to see which robot is more certain about its location, and then, based on the result of the comparison, we decide how to exchange information between two robots. Figure 3.3 illustrates this scenario where both robots have already obtained some knowledge about their poses by landmark detection. When robot A and robot B detect each other, they need to compare their beliefs.

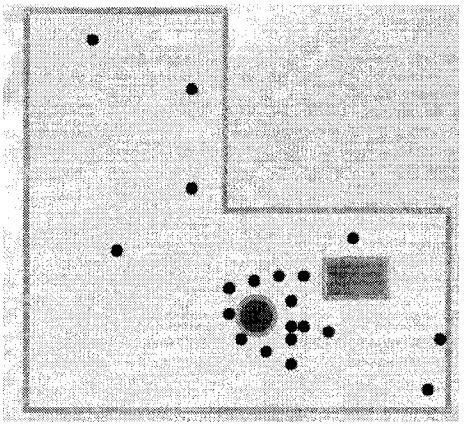
In order to compare with the beliefs of two robots, we need to evaluate the degree of uncertainty within the belief of each robot. To evaluate the uncertainty, our approach employs the entropy. In the next section, the background knowledge of information theory will be presented.



(a)

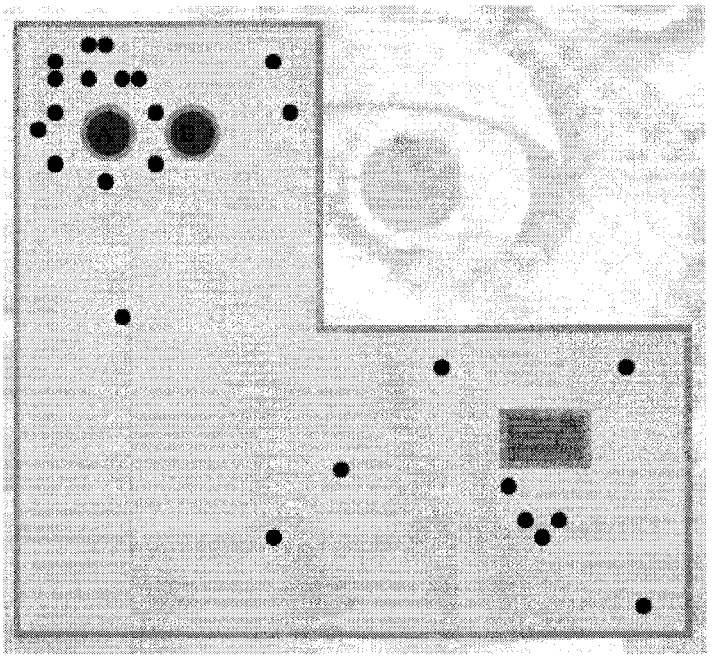


(b)

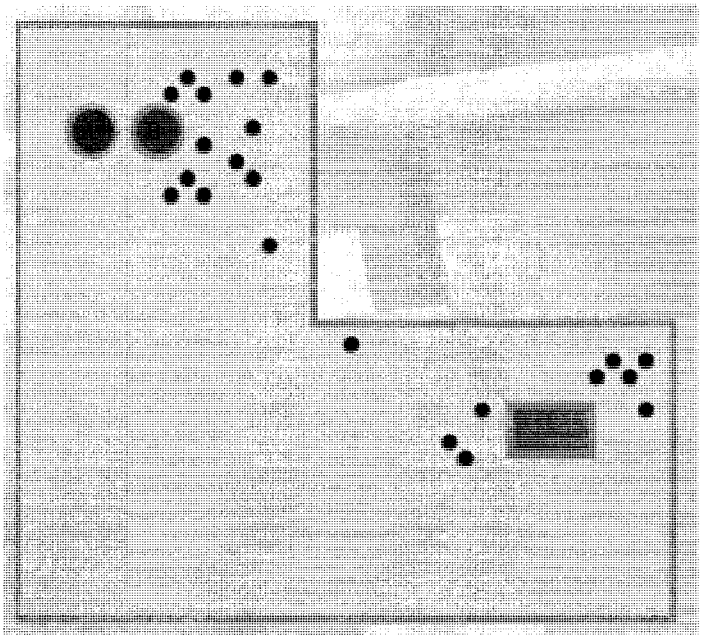


(c)

Figure 3.2: Two robots detect each other with information exchange: (a) Robot *A*'s belief becomes more certain by detecting a landmark; (b) The belief of robot *B* is still very much uncertain; (c) The result belief of robot *B* after refined by robot *A*' belief. (Robot *A* and *B* are in the same environment)



(a)



(b)

Figure 3.3: Illustration of scenario three: (a) Robot A's belief becomes more certain by detecting a landmark; (b) The belief of robot B also becomes more certain through the landmark detection. They first need to compare their beliefs, and then decide how to exchange their information. (Robot A and B are in the same environment)

3.3 Background Knowledge of Information Theory

Information theory [48] is a branch of applied mathematics and engineering involving the quantification of information. Information theory is generally considered to have been founded in 1948 by Claude Shannon in his seminal work: A Mathematical Theory of Communication [48]. Information theory is a broad and deep mathematical theory, with equally broad and deep applications. Information theory is based on probability theory and statistics.

A key measure of information that comes up in the theory is known as information entropy [72], which is usually expressed by the average number of bits needed for storage or communication. Intuitively, entropy quantifies the uncertainty involved in a random variable. For example, a fair coin flip will have less entropy than a roll of a die. Information entropy quantifies the information contained in a message, usually in bits or bits/symbol. The choice of logarithmic base in the following formula determines the unit of information entropy that is used. The most common unit of information is the bit, based on the binary logarithm. Other units include the nat, which is based on the natural logarithm.

The entropy H of a discrete random variable X that can take on possible values $\{x_1...x_n\}$ is defined as:

$$H(X) = \mathrm{E}_X[I(x)] = -\sum_{x \in X} p(x_i) \log_b p(x_i) \quad [48]\ (1),$$

where $I(x)$ is the information content or self information, which is the entropy contribution of an individual message; $p(x_i) = \Pr(X = x_i)$ is the probability function of outcome x_i, and b is the base of the logarithm used. Possible values of b are 2, e, and 10. The unit of the entropy H is bit for $b = 2$, nat for $b = e$, and digit for

$b = 10$. An important property of information entropy is that it has maximized uncertainty when all the random variables in the variable set are all having the equal probability.

3.4 Explanations of Proposed Method (Continue I)

As we described the last scenario in section 3.2, if both robots have already obtained some knowledge about their poses relative to the environment during the detection time, we need to evaluate the degree of uncertainty for each robot's belief. In order to evaluate the degree of uncertainty, our approach makes use of formula (1) to calculate how much uncertainty along with each robot's belief, and then uses these entropy values for the comparison. For instance, we have two robots A and B in the environment. At the time of detection, the entropy value of robot A's belief is H_A, and the entropy value of robot B's belief is H_B. Obviously, there are three different situations if one compares H_A with H_B.

(a) The first situation is that $H_A > H_B$, which indicates that robot A's belief is less certain than robot B's. Because the belief of robot B contains more information than robot A does, the robot B will use its belief to refine the belief of robot A.

(b) The second situation is that $H_A < H_B$, which means that robot A's belief is more certain than robot B's. The robot A will use its more certain belief about its pose to help refine robot B's belief.

(c) The last situation is that $H_A = H_B$, which shows that robot A and robot B have the same degree of uncertainty for their beliefs. Because both robots have the same degree of uncertainty for their beliefs in the last circumstance, there is no

information exchange between robots.

In order to apply the formula of entropy, we must know the probability density values of all the discrete random variables in the dataset. Within the context of MCL based multiple robots localization, the belief of a robot is represented by a collection of weighted particles. Because each particle in the particle set has two dimensional coordinates x and y, we may refer each particle as a two dimensional random variable. Consequently, we have a set of discrete random variables in the two dimensional space to represent the belief of a robot. Then the problem of calculating the probability density values of the particle set will be transformed to calculate the probability density values of a collection of discrete two dimensional random variables. There are many existing approaches [20, 46] addressing the problem of extracting the probability density values. One of the commonly used approaches to solve this problem is density estimation [49].

Our approach will exploit the technique of density estimation for calculating the probability density values. In the next section, the background knowledge of density estimation including idea of density estimation, kernel density estimation [37], and multivariate kernel density estimation [47] will be presented.

3.5 Background Knowledge of Density Estimation

In probability and statistics, density estimation is the construction of an estimate, based on observed data, of an unobservable underlying probability density function. The unobservable density function is thought of as the density according to which a large population is distributed; the data are usually thought of as a random sample from that population. A variety of approaches to density estimation are used, including kernel density approximation [37] and a range of data clustering techniques.

In statistics, kernel density estimation is a way of estimating the probability density function of a random variable. If $x_1, x_2, \ldots x_N$ are random variables in a sample set, then the kernel density approximation of its probability density function is:

$$p(x) = \frac{1}{Nh} \sum_{i=1}^{N} K(\frac{x - x_i}{h}) \quad [47] (2),$$

where N is the number of random variables, K is some kernel [73] which is a weighting function used in non-parametric estimation techniques, and h is the smoothing parameter which attempts to capture important patterns in the data while leaving out noise.

In our case, we need to calculate probability density of all the particles in the particle set. Because each particle is a two dimensional random variable which includes coordinates x and y, we must be able to estimate multivariate densities. Consider a d dimensional random vector $x = (x_1, x_2, \ldots x_d)^T$ where $x_1, x_2, \ldots, x_d$ are one dimensional random variables. Drawing a random sample of size N in this setting means that we have N observations for each of the d random variables, $x_1, x_2, \ldots, x_d$. Suppose that we collect the ith observation of each of the d random variables in the vector x_i:

$$x_i = (x_{i1}, x_{i2}, \ldots, x_{id}), i = 1, 2, \ldots, n \quad [47] (3),$$

where x_{ij} is the ith observation of the random variable x_j. Our goal now is to estimate the probability density of $x = (x_1, x_2, \ldots x_d)^T$, which is just the joint probability density function p of the random variables $x_1, x_2, \ldots, x_d : p(x) = p(x_1, \ldots x_d)$.

From the above one dimensional case we might consider adapting the kernel density estimator to the d dimensional case, and write in the following form:

$$p(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h^d}K(\frac{x-xi}{h}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h^d}K(\frac{x_1 - x_{i1}}{h},...,\frac{x_d - x_{id}}{h}) \quad [47]\ (4),$$

where K denotes a multivariate kernel function operating on d arguments. We assume that the smoothing parameter h is the same for each component. What form should the multiple dimensional kernels take on? The easiest solution is to use a multiplicative kernel $K(u) = K(u_1) * ... * K(u_d)$ [47], where K denotes a univariate kernel function. In our case becomes:

$$p(x) = \frac{1}{n}\sum_{i=1}^{n}\{\prod_{j=1}^{d}\frac{1}{h}K(\frac{x_j - x_{ij}}{h})\} \quad [47].\ (5)$$

To get a better understanding, let us consider the two dimensional case where $x = (x_1, x_2)^T$. In this case, the multivariate kernel density function becomes:

$$p(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h^2}K(\frac{x_1 - x_{i1}}{h}, \frac{x_2 - x_{i2}}{h}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h^2}K(\frac{x_1 - x_{i1}}{h})K(\frac{x_2 - x_{i2}}{h}) \quad [75].\ (6)$$

Each of the n observations is of the form (x_{i1}, x_{i2}), where the first component gives the value that the random variable x_1 takes on at *ith* observation and the second component does the same for x_2. Notice that we get a contribution to the sum for observation i only if x_{i1} falls into the interval $[x_1 - h, x_1 + h)$ and if x_{i2} falls into the interval $[x_2 - h, x_2 + h)$. If even one of the two components fails to fall into the respective interval then one of the indicator functions takes the value 0 and

consequently the observation does not count [75].

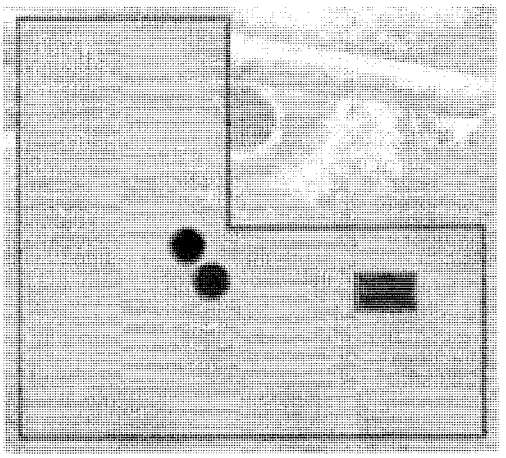
3.6 Explanations of Proposed Method (Continue II)

According to the previous explained formula (5) of multivariate kernel density estimation and formula (1), our approach is capable of calculating the entropy of a robot's belief. The entropy value reflects the uncertainty about a robot's knowledge of its pose relative to its surroundings. For example, if two robots A and B detect each other after observing the landmarks, they will use above described method to compute entropies of their beliefs. If robot A's entropy is smaller than robot B's, which indicates that robot A' belief is more certain than robot B's, and then robot A will use its belief to refine robot B's belief, and vice versa. In doing so, the robot whose belief is uncertain will benefit from the process of this information exchange. Accordingly, the localization process of the whole system will be accelerated.

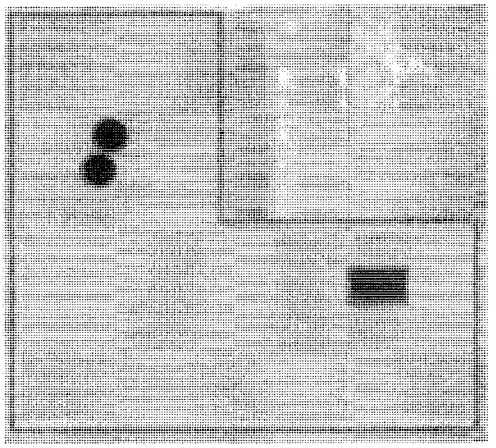
Because the longer the passage of time is since the last detection with another robot, the more chance the robot loses its position or becomes uncertainty again. To avoid exchanging unnecessary information between two robots, we add one constrain to our method. In our approach, if both robots have already exchanged their beliefs with each other, the previous defined status variable of landmark detection in section 3.2.2 for both robots will be set to false, which indicates that they can not exchange their beliefs again until the next time after one of the robots observes the landmark or both robots detect the landmarks. By doing so, our method lets the robot exchange its belief with another robot only after the robot's significant movement (e.g. detects a landmark) from the last detection of another robot. In Figure 3.4 (Because we do not need to show the belief of the robot, we get rid of all the particles in this Figure), if robot A has already exchanged its belief during the detection with robot B, the robot A will not exchange its belief with robot B in Figure 3.4(b) until after the next time robot A observes a landmark or both robots detect the landmark. For the sake of avoiding

unnecessary information exchange, a robot would exchange its belief after significant movement from last detection of another robot. Therefore, our approach handles with the delayed information integration in a more appropriate way.

The above sections described the method for multi-robot localization. Since each robot in our system performs single mobile robot MCL, the MCL algorithm is the basis of our approach. In the following section, we are about to explain the main components which are included in MCL.



(a)



(b)

Figure 3.4: Illustration of additional constrain: (a) Robot *A* just exchanged its belief with Robot *B*; (b) Robot *A* will not exchange its belief with robot *B* until after the next time of significant movement (e.g. landmark detection) for one of the robots or both.

3.7 Components of MCL

As mentioned in Chapter2, our proposed method is based on the MCL which consists of three key components: motion model, measurement model, and importance sampling. In the following, we explain these components in detail.

3.7.1 Motion Model

The motion model is one of the important components in the field of probabilistic robotics for implementing the Bayes filter algorithms. Motion model is also called state transition probability $p(x_t \mid x_{t-1}, u_t)$, which plays an essential role in the step of the Bayes filter [60]. This section provides detailed explanation of probabilistic motion models as they are used in our approach.

Kinematics is the calculus describing the effect of control actions on the configuration of a robot [7]. We entirely focus on mobile robot kinematics for robots operating in planar environments, whose kinematical state is represented by three variables referred to as pose [60]. The pose of a mobile robot operating in a plane comprises the two dimensional coordinates relative to its external coordinate frame, along with its orientation. We denote the former as x and y, and the latter by θ. Pose without orientation will be called location. The probabilistic kinematical model or motion model plays the role of the state transition model in mobile robotics. This model is the conditional probability density $p(x_t \mid x_{t-1}, u_t)$ as we explained in chapter 2. Here x_t and x_{t-1} are both robot poses, and u_t is a motion command. This model describes the posterior distribution over kinematical states that a robot supposes to be when executing the motion command u_t at x_{t-1}. In our implementations, u_t is provided by a robot's odometry. Figure 3.5 shows two examples to illustrate the

motion model for a mobile robot operating in a planar environment. In both cases, a robot's initial pose is x_{t-1}. The probability distribution $p(x_t \mid x_{t-1}, u_t)$ is represented by the shady area. It means that the darker the area, the more likely the true pose of a robot will be. In Figure 3.5(a), a robot moves forward some distance, which might accumulate error for both translation and rotation as indicated. Figure 3.5(b) shows the result distribution of a more complicated motion command, which causes to a larger spread of uncertainty compared with the first case.

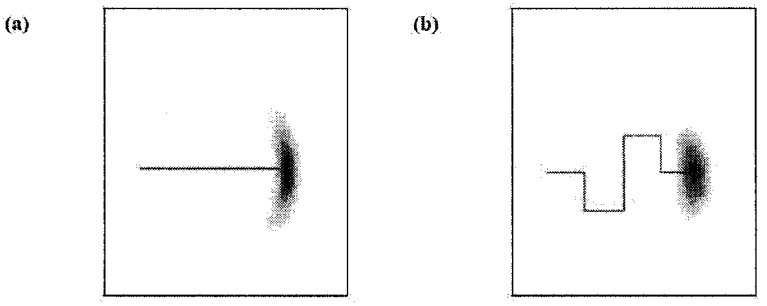


Figure 3.5: The examples of the motion model. [60]

As mentioned above, we use odometry measurements as the control commands for calculating the posteriors over poses. This will lead us to a specific probabilistic motion model referred to as odometry motion model which uses odometry measurements for motion controls.

Let us first define the format of our control information. At time t, the pose of a robot is modeled by the random variable x_t. We need to use the robot's odometry to estimate this pose. Nevertheless, due to the slippage and drift there is no fixed coordinate transformation between the coordinates used by the robot's internal odometry and its external world coordinates. The key idea of odometry motion model [60] is using the relative motion information measured by the robot's internal odometry. To extract relative odometry, u_t is transformed into a sequence of three steps: a rotation, followed by a straight line motion, and then another rotation [55].

This decomposition is illustrated in Figure 3.6. The first turn is called δ_{rot1}, the translation δ_{trans}, and the second rotation δ_{rot2}. Each pair of positions (the robot has a starting position and an ending position for each control command) has an unique parameter vector $(\delta_{rot1}\delta_{trans}\delta_{rot2})^T$, these parameters are adequate to rebuild the relative motion between two positions. Therefore, δ_{rot1}, δ_{trans}, and δ_{rot2} supply sufficient information of the relative motion encoded by the odometry data. The probabilistic odometry motion model assumes that these three parameters are corrupted by independent noise [55].

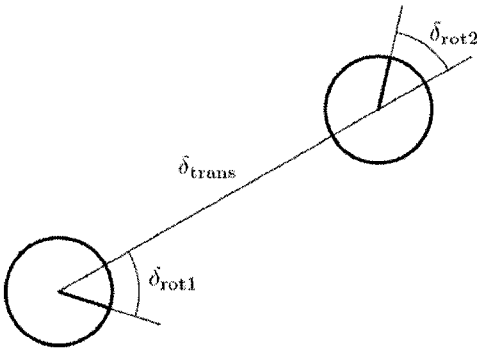


Figure 3.6: The odometry motion model. [60]

There are two forms of algorithm for computing the probability $p(x_t \mid x_{t-1}, u_t)$: one is closed form calculation, another is sampling form. Because MCL uses samples to represent a robot's pose, our method makes use of sampling form algorithm for computing this probability density. The algorithm *sample_motion_model_odometry* [60] showed in Table 3.2 implements this sampling approach. It accepts an initial pose x_{t-1} and an odometry data u_t as inputs, and outputs a random pose at time t drawn distributed according to $p(x_t \mid x_{t-1}, u_t)$. The variables, α_1 to α_4, are robot specific error parameters which specify the noise in robot motion. They model the accuracy of the robot (The less accurate a robot, the larger these parameters). The parameters α_2

and α_3 are used to control translation error, while parameters α_1 and α_4 are used to control angular error.

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

2: $\delta_{\text{rot1}} = \text{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta}$

3: $\delta_{\text{trans}} = \sqrt{(\bar{x} - \bar{x}')^2 + (\bar{y} - \bar{y}')^2}$

4: $\delta_{\text{rot2}} = \bar{\theta}' - \bar{\theta} - \delta_{\text{rot1}}$

5: $\hat{\delta}_{\text{rot1}} = \delta_{\text{rot1}} - \text{sample}(\alpha_1 \delta_{\text{rot1}} + \alpha_2 \delta_{\text{trans}})$

6: $\hat{\delta}_{\text{trans}} = \delta_{\text{trans}} - \text{sample}(\alpha_3 \delta_{\text{trans}} + \alpha_4 (\delta_{\text{rot1}} + \delta_{\text{rot2}}))$

7: $\hat{\delta}_{\text{rot2}} = \delta_{\text{rot2}} - \text{sample}(\alpha_1 \delta_{\text{rot2}} + \alpha_2 \delta_{\text{trans}})$

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

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

10: $\theta' = \theta + \hat{\delta}_{\text{rot1}} + \hat{\delta}_{\text{rot2}}$

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

Table 3.2: The sample odometry motion model algorithm. [60]

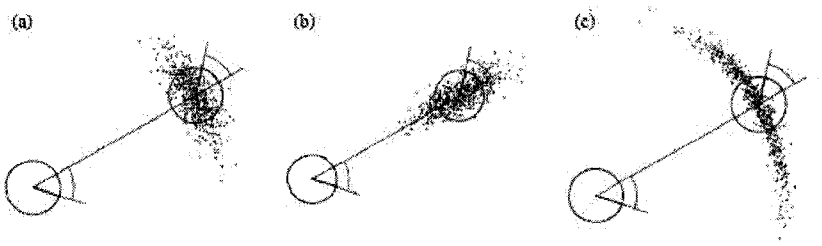


Figure 3.7: Sampling from the odometry motion model. [60]

Figure 3.7 shows examples of sample sets generated by algorithm of sample_motion_model_odometry. The sample set in Figure 3.7(a) is a typical one,

whereas the ones shown in Figure 3.7(b) and 3.7(c) indicate unusually large translation and rotation errors respectively.

Figure 3.8 illustrates the odometry motion model in action. The solid line displays the actions taken by a robot, and the samples represent the robot's belief at different pose. This figure shows that how the uncertainty grows as the robot moves. The samples are spread out in an increasingly large space without perceiving the external world.

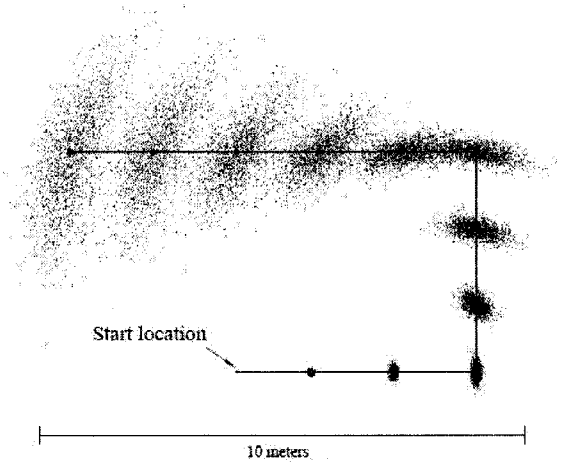


Figure 3.8: Sampling approximation of a robot's belief. [60]

3.7.2 Measurement Model

Other than probability motion model, there is another specific model in probabilistic robotics used in MCL, called measurement models [60]. The probability measurement models describe the formation process by which sensor measurements are generated in the physical world. Nowadays, robots can be equipped with many different sensors such as tactile sensors, range sensors, or cameras. The specifics of the measurement models depend on different sensors. For example, laser range finders are best modeled by using a laser beam in order to determine the distance to a reflective object. Probabilistic robotics explicitly models the noise in sensor

measurements. Formally, the measurement model is defined as a conditional probability distribution $p(z_t \mid x_t, m)$, where x_t is the robot pose, z_t is the measurement at time t, and m is the map of the environment. This conditional probability distribution $p(z_t \mid x_t, m)$ is used to specify the measurements z_t are generated from the environmental state x_t according to the given map of the environment m. The measurement model according to our specific robot platform will be given in the next chapter.

3.7.3 Resampling

Another important component of MCL is known as resampling or importance sampling [10, 14]. The key idea of resampling is that choosing M random numbers and selecting those particles that correspond to these random numbers, the distribution of selected particles is according to the probability proportional to the particles' weights. The step of resampling transforms a particle set of M particles into another particle set by incorporating the importance weights into the resampling process, the distribution of the particle change: Whereas before the resampling step, they distributed according to $\overline{bel}(x_t)$, after the resampling they are approximately accordingly to the posterior $bel(x_t)$. The resampling procedure is a probabilistic implementation of the Darwinian idea of survival of the fittest [60]. It refocuses the particle set to regions in the state space with high posterior probability. By doing so, it focuses the computational resources to regions in the state space where most relevant.

3.8 Summary

In this chapter, we propose a new information exchange mechanism for collaborative multi-robot localization. We also propose a new scheme to calculate how

much information is contained in a robot's belief by using information theory. According to the analysis, it is expected that our approach can avoid unnecessary information exchange whenever one robot perceives another robot. On the other hand, it is also expected that this approach does allow information exchange between detecting robots which contributes positively to the localization process, hence, improving the effectiveness and efficiency of multi-robot localization. In the next chapter, we will demonstrate the above analysis. The detailed implementations of the above proposed approach and the experimental results will also be presented in the following chapter.

Chapter 4
Implementation and Experimental Results

In this chapter, we will present the implementation details of our experiments in section 4.1, which include hardware platform and its setup, programming environment, and implementation of MCL algorithm on iRobot Create and iRobot Discovery. The experimental results will be given in section 4.2.

4.1 Implementation Details

4.1.1 Hardware Platform

In our experiment, the vacuum cleaning robots Roomba [76] will be used. Roomba is an autonomous robotic cleaner created by iRobot Corporation. The Roomba was first released in 2002 with updates and new models released every year since. Compared to other vacuum cleaners, the typical Roomba robotic vacuum cleaner is very inexpensive at under $300 for even the most expensive Roomba and $150 for the least expensive. The iRobot's Roomba vacuum represents the growing ubiquity of robotics perhaps better than any other single platform. Over two million Roombas clean floors in homes and businesses [64]. The platform has become a standard for task based, low cost robotics available for research and education. Roomba can be programmed and accessed without any modification.

We use the third generation of Roomba in our experiment, which includes many more improvements than the first and second generation. In addition to dirt sensor, these models include a home base dock for self-charging, a remote control, and the most important for robotic fans, a serial port. The current generation of Roomba is organized in three sections [30]:

Sensor front: All of the sensors such as bump, wall, cliff, and home base contacts are up front. In fact, almost all the sensors are mounted on the movable front bumper. This movable bumper not only enables a way to measure contact which gives triggers a switch, but also absorbs shock to minimize damage. The Roomba firmware is designed to always travel forward, so it places its most sensitive parts forward.

Motor middle: The main drive motors, vacuum motors, vacuum brushes, side cleaning brush, and battery are all in the center. This kind of physical structure makes Roomba very stable when moving.

Vacuum back: Just like a normal vacuum cleaner, the entire back of Roomba contains the vacuum and vacuum bag for holding dirt.

As described above, the Roomba equips with many useful sensors. Figure 4.1 shows a Roomba along with a variety of sensors. The Roomba navigates mainly by its mechanical bump sensors, infrared wall sensors. For detecting dangerous conditions, it also has infrared cliff detectors and wheel drop sensors. In the following, we describe some sensors which are used in our experiment.

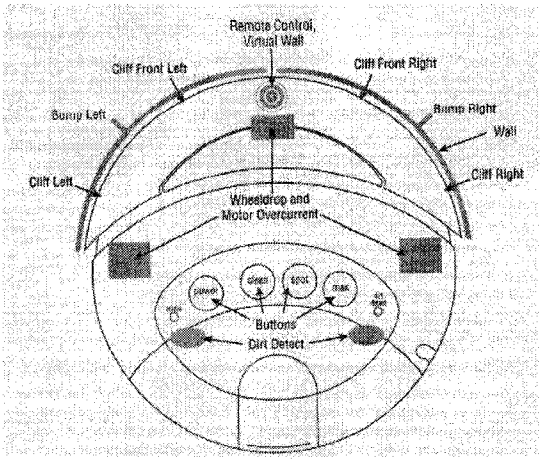


Figure 4.1: Location of Roomba sensors. [30]

Bump Sensors: Roomba has two bump sensors on the front, located at 11 o'clock and 1 o'clock positions. The spring loaded front bumper moves to trigger one or both of these sensors. Each is implemented as an optical interrupter. In the case of Roomba's bump sensor, the interrupter is a small plastic arm connected to the bumper.

Infrared Sensors: There are six infrared sensors on the Roomba, all on the front bumper. Four of these facing to the ground are the cliff sensors, and another facing to the right is the wall sensor. These five sensors work much like the bump sensors, because there is an LED emitter and a photo detector looking for the LED's light. But unlike the interrupter based sensor, these sensors are looking for the reflected light of the LED. For the cliff sensors, they are looking for light reflected from the floor. For the wall sensor, it is looking for a wall. The last infrared sensor is the remote control, virtual wall or docking station sensor which can be found at the 12 o'clock position on the bumper. This sensor works just like any other remote control sensor for consumer electronics. In our experiments of real robot, we use virtual wall sensor to perceive another robot.

Internal Sensors: The most commonly used internal sensors are the odometer sensor. We are able to retrieve the distance and angle values from this sensor. The distance is obtained from the optical interrupter sensor on the wheels. The value comes from counting the number of beam interruptions caused from the toothed interrupter disc. The firmware specification gives a distance resolution of 1 mm. Although the distance value is a straightforward measurement, the angle value is an odometry difference. Roomba has a distance sensor on each wheel, and the angle in the sensor data is the difference in the distance traveled by each wheel. This difference describes a rotation around the center point between the two wheels. The wheel drop sensors have a micro switch which detects when the wheel is down. These wheel drops are equivalent to cliff detection since they are indicating that the Roomba is in some unforeseen situation and should stop from its current task. The last collection of internal sensors is the power measurement sensors.

In order to connect and communicate with Roomba, we use a device called RooTooth [30]. RooTooth provides a cable less solution for controlling the Roomba. It also provides Bluetooth capabilities to Roomba and allows us to connect and communicate with it through any Bluetooth enabled devices over Bluetooth's Serial Port Profile (SPP). RooTooth is ideal for wirelessly interfacing the Roomba to common Bluetooth enabled devices such as PCs, laptops, and cell phones [30]. Communication with Roomba occurs through a virtual COM Port created on the device by using Bluetooth's SPP, which allows for serial communications wirelessly. Any of the programs which can talk to the serial port is capable of sending commands to the Roomba as well as receive information from it. The RooTooth is designed for accommodating Class1 or Class2 Bluetooth radio modem serial modules at 2.4GHz frequency. Figure 4.2 shows a RooTooth adapter. RooTooth simply plugs into Roomba's expansion port and enables this vacuum cleaner to take advantage of the Roomba Open Interface (ROI) which will be described in section 4.1.2. Therefore, it offers the most flexible and stable way of communication and it also brings connection quality over distance dropping slowly.

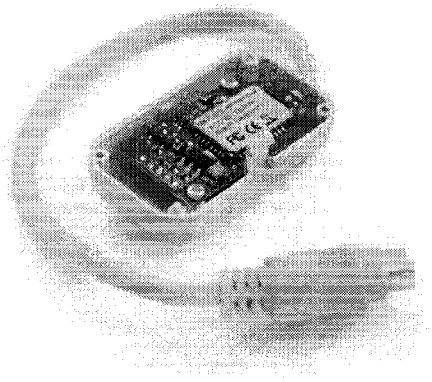
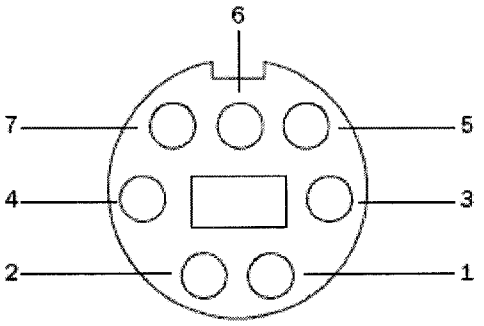


Figure 4.2: RooTooth Bluetooth Roomba adapter. [30]

4.1.2 Programming Environment

The Roomba manufactured after October, 2005 contains an electronic and software interface that allows us to control or modify Roomba's behavior and remotely monitor its sensors. This interface is called the iRobot Roomba Open Interface or iRobot ROI [77] (The detailed information of iRobot ROI will be provided in Appendix A). The iRobot ROI is a serial protocol that allows users to control a Roomba through its external serial port, called Mini-DIN connector shown in Figure 4.3. The ROI includes commands to control all of Roomba's actuators such as motors, lights, and speaker and also request sensor data from all of Roomba's sensors. Thus, much of the low level hard work dealing with motors and sensors has been taken care of inside the Roomba itself. It offers an almost complete view of the Roomba's internals. It abstracts certain functions, making then easier to use. By using the ROI, users can add functionality to the normal Roomba behavior or they can create completely new operating instructions for Roomba.



Pin	Name	Description
1	Vpwr	Roomba battery + (unregulated)
2	Vpwr	Roomba battery + (unregulated)
3	RXD	0 – 5V Serial input to Roomba
4	TXD	0 – 5V Serial output from Roomba
5	DD	Device Detect input (active low) – used to wake up Roomba from sleep
6	GND	Roomba battery ground
7	GND	Roomba battery ground

Figure 4.3: Roomba ROI connector Mini DIN. [30]

When using the ROI, Roomba can exist in one of five states. These states represent both how Roomba behaves and how it responds to ROI commands. Actions by Roomba can also change the state. Some of the ROI commands are only used to select the suitable state because some commands only work under certain state. Herewith the followings are list of five states [30]:

(a) **Off:** Roomba responds to no commands over the ROI, but can be woken up and put into the on state by using the power button.

(b) **On:** Roomba is awake and is awaiting a START command over the ROI. In this state Roomba is able to work normally through its button or remote control. The only way out of this state through the ROI is using the START command.

(c) **Passive:** Roomba has received the START command. In this state sensors can be retrieved, but no control of the robot is executed over the ROI. The Roomba buttons work as usual. The state is used to monitor the Roomba as it goes about its work. The usual step from this state is to send the CONTROL command to enter safe mode.

(d) **Safe:** Roomba has received the CONTROL command from passive state or the SAFE command from full state. Everything that can be done in passive state is still possible, but now Roomba can be controlled. The buttons on Roomba no longer change the robot's behavior; instead their states are reflected in the Roomba sensors data. All commands are now available, but a built-in safety feature exists to help us not damage the Roomba. This safety feature is activated if Roomba detects the following: a cliff is encountered while moving forward, any wheel drops, and the charger is plugged in.

(e) **Full:** If Roomba receives a FULL command while in safe state, it will switch to this state. This state is the same as safe mode except that the safety features is

turned off. To get out of this state, send the SAFE command. Sending the POWER button command will put Roomba into off state.

The Roomba changes from one state to the next depending on either ROI commands or external events. Figure 4.4 shows the Roomba ROI state change diagram.

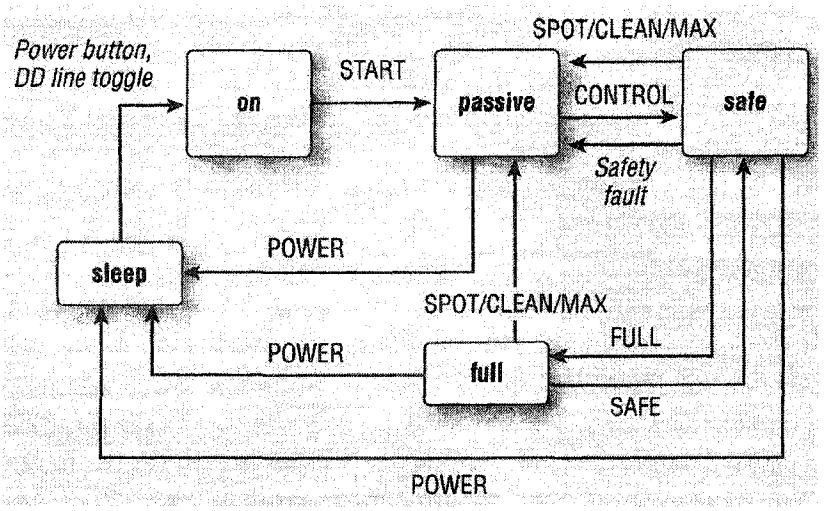


Figure 4.4: Roomba ROI state diagram. [30]

The ROI protocol is quite rudimentary. The protocol is a simple byte oriented binary serial protocol. It would be a lot easier if there is a library to help us to make things work. The *RoombaComm API* [78] is just such an encapsulation of the ROI binary commands into the more easily accessible Java classes. RoombaComm API is a Java library for communicating and controlling the Roomba. It works on any operating system that RXTX [79] (serial and parallel I/O libraries supporting Sun's CommAPI) supports including Windows, Linux, and Mac OS. RoombaComm API is also used to make coding easier. The goals of this library is to provide full access to the entire ROI protocol and a collection of high level functionality on the top of the ROI protocol, creating a library that is as cross platform as possible.

Because we need to use a lot of the graphics and animations in our experiments, we then decide to use *Processing* [80]. The Processing is a free open source programming language and environment for people who want to write graphical programming quickly and easily. In many ways, processing is a descendant of the Logo programming language. Both are visually focused and provide a number of functions to make drawing graphics and building animations easier. However, Processing can do much more. It can operate in 3D, work with video and sound, perform physical simulations, and do many other things. It is continuously being expanded and improved through libraries created by anyone with good ideas. Processing is implemented in Java. The Processing language is no different from Java at all; it just removes the complexity from Java. Therefore, Processing is a kind of Java IDE. It enables us to create, compile, and run dynamic graphical programs. The most important thing for us to do the experiment is that Processing is fairly easy to use the full Java class library or wrap up any other Java class into a Processing library. This is what has been done to allow us to make use of RoombaComm in Processing.

4.1.3 Implementations of Motion Model and Measurement Model

We have already illustrated how Roomba works and how to interact with Roomba. In this section, we are about to describe the implementations of motion model and measurement model for MCL using to Roomba.

As described in section 3.7.1, the motion model of MCL can be implemented by using sampling algorithm sample_motion_model_odometry. In order to use this algorithm, one needs to make use of the odometry measurements as the basis for calculating the robot's motion over time. Odometry is commonly obtained by integrating wheel encoder information. Most of the commercial robots provide odometry using kinematic information. In our experiment, we can access odometry reading of Roomba through RoombaComm API. In doing so, we can obtain the

odometry information for Roomba such as distance traveled, angle turned. Subsequently, this information is used for computing probability distribution $p(x_t \mid x_{t-1}, u_t)$ which is the motion model.

Other than motion model, MCL has another important component which is measurement model. Roomba is equipped with bumper sensors and infrared sensors to perceive its surroundings. Compared with other robots which equip with more powerful sensors, Roomba only has such limited sensors to detect the external environment. In order to implement the measurement model, we need to distinguish two types of detections: (a) a robot detects the landmark such as wall or any other obstacles in the environment; (b) detection between two robots. We use Roomba's bumper sensors to measure the first type of detection. If the bumper sensors return readings, it indicates that Roomba has detected any of the landmarks. For the sake of detecting between robots, we put a virtual wall (a standard IR remote transmitter) on the top of one robot. If another robot's virtual wall sensor has reading, it means that there must be a robot close by itself. On condition that a robot has the first type of detection, we assign the high probability (high importance) to particles which are close to the landmarks in the environment, and low probability (low importance) to the rest of the particles. By doing so, these weighted particles can be used for the step of importance sampling or resampling described in section 3.7.3. In the event a robot receives the second type of detection, our algorithm will decide if it is necessary to exchange information with another robot.

In section 4.1, we have described our experimental environment, including hardware and software. Since our approach is based on MCL, we have also explained the implementation details about motion model, measurement mode using Roomba. In the next section, the detailed experimental results will be presented.

4.2 Experimental Results

In this section we present experiments conducted with both real and simulated robots. The central question driving our experiments is: to what extent can cooperative multi-robot localization improve the localization quality through our proposed approach, when compared to the single mobile robot localization.

In the following experiments we use a tool which developed by ourselves. Under the help of this tool, we can test our proposed method in a variety of scenarios. We can also measure the distance traveled by a robot and elapsed time used by a robot to localize itself within the given map of the environment.

In the real robots experiments, our approach was tested using two Roombas (iRobot Discovery and iRobot Create) shown in Figure 4.5. The iRobot Discovery is a third generation of Roomba cleaner. The iRobot Create is a hobbyist robot based on the Roomba platform (a non-vacuum Roomba). In order to evaluate the benefits of multi-robot localization in more complex scenarios, we additionally performed our experiments in simulated environments. These experiments are described in Section 4.2.3.

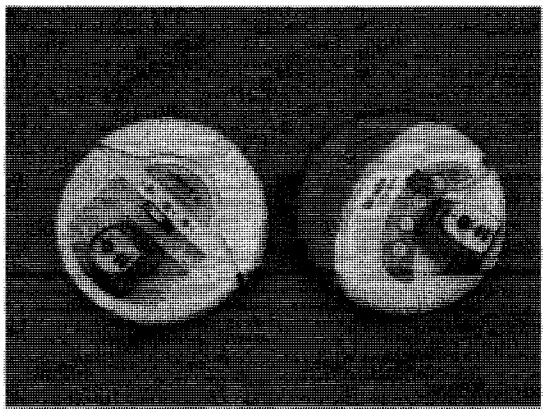


Figure 4.5: Two real robots for our experiment: iRobot Create (left) and iRobot Discovery (right).

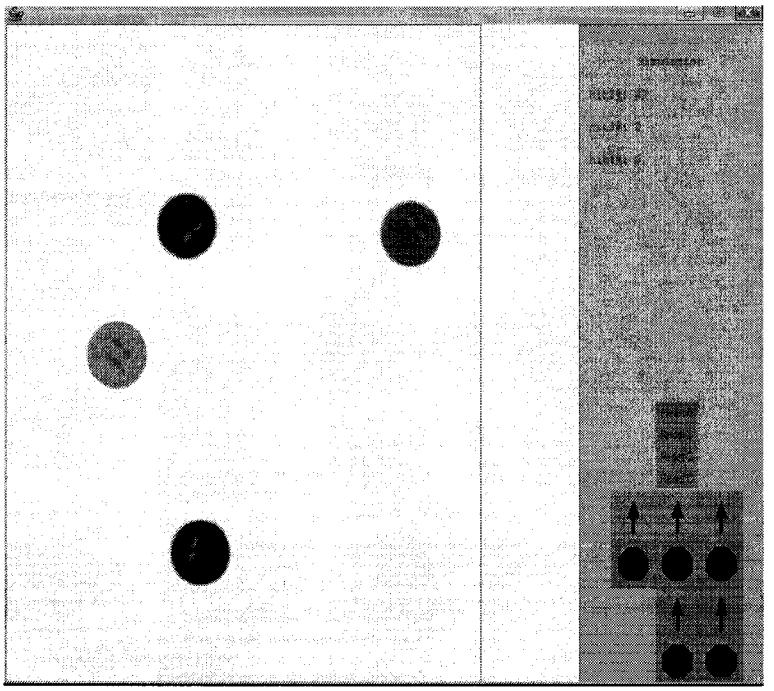
4.2.1 Verification of an Assumption

In section 3.2.1, we made an assumption of our proposed method which is that there is a remote chance that more than two robots detect each other simultaneously, which happens rarely and will be ignored when it happens. In this section, we will verify this assumption through our designed experiments.

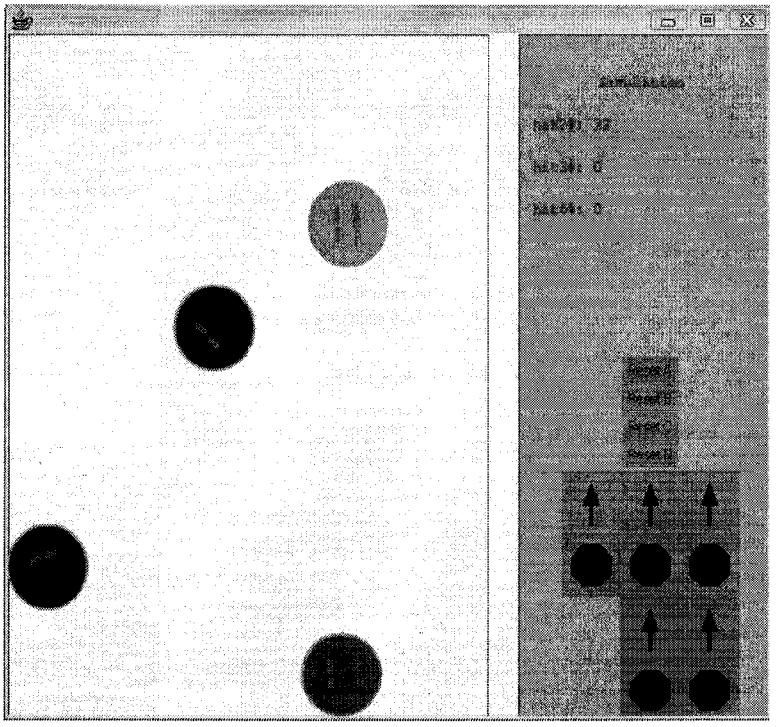
In this simulation experiment we use four robots. The task of these robots is to perform random exploration in a rectangular free space shown in Figure 4.6. In order to support this assumption, two maps of different sizes are used. During the experiment, our designed tool is able to count how many times two robots detect each other at the same time, how many times three robots detect each other concurrently, and how many times four robots detect each other simultaneously.

Figure 4.6(a) shows the first run, we use a 488 pixels $\times$ 610 pixels map, and the radius of each Roomba is 30 pixels. After 15 minutes, the simulation tool shows that there are 246 detections between two robots at the same time. The situations where three robots detect each other at the same time and four robots detect each other at the same time never happen. Figure 4.6(b) shows the second time run, we change the map to smaller size, which is 366 pixels $\times$ 488 pixels. After the same duration as the previous experiment, there are 438 detections between two robots at the same time. The detections among three robots and four robots at the same time still never happen. The experimental results demonstrate that there is a remote chance that more than two robots detect each other simultaneously is a reasonable and practical assumption.

Since we have this assumption, the following experiments which include real robot experiments and simulation robot experiments will focus on the situations where only two robots are involved.



(a)



(b)

Figure 4.6: Simulation experiments for verifying one of the assumptions. (a) Experiment in a bigger map (488 pixels $\times$ 610 pixels). (b) Experiment in a smaller map (366 pixels $\times$ 488 pixels).

4.2.2 Experiments Using Real Robots

In this section we present experiments conducted with real robots. In the first experiment we test the case of single mobile robot localization using MCL in two maps with different size. The purpose of this experiment is that we are able to use the experimental results to compare with the experimental results of our proposed method. As the result, we will see that if the localization process of the system can benefit from our proposed approach. In the second experiment we test the case of multi-robot localization using our proposed approach in the same maps as used in previous experiment of single mobile robot localization. The experimental results and evaluations will also be given in this section.

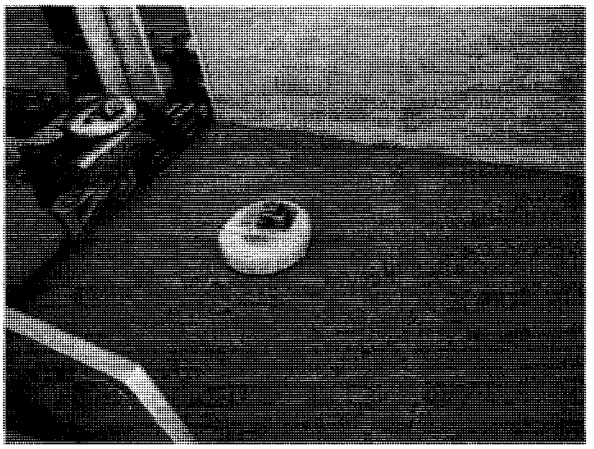
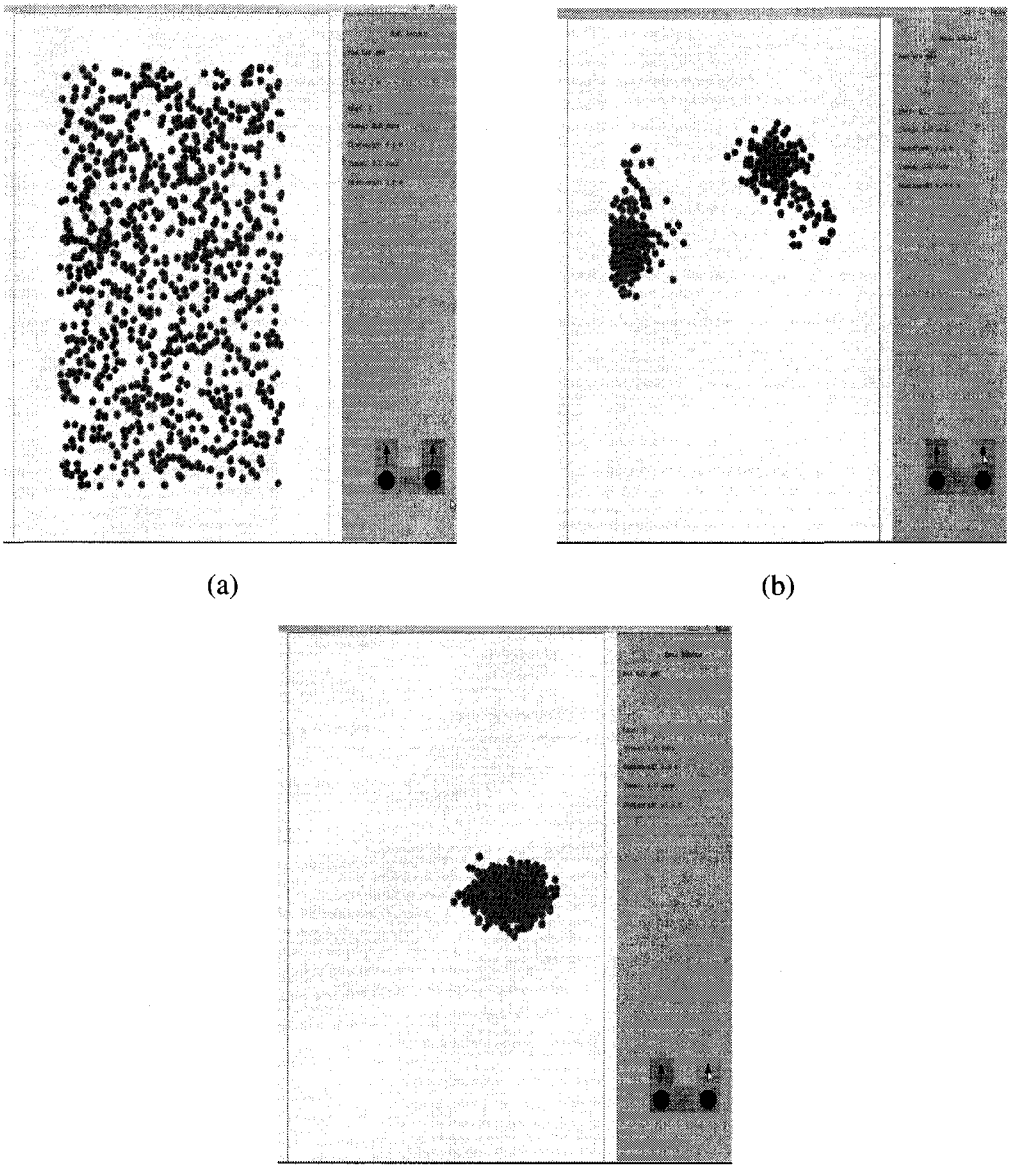


Figure 4.7: Experimental environment for single mobile robot localization.

(a) Experiment of Single Mobile Robot Localization

In this experiment we use an iRobot Create to perform global localization in the rectangular free space shown in Figure 4.7. The environment is particularly challenging for single robot system since its external environment is symmetry. We test this case in two maps with different size. In the first run, we use a 213.5 cm $\times$152.5 cm map. We use 1000 particles to represent the belief of this robot. Initially,

all the particles are spread over according to the uniform distribution shown in Figure 4.8(a). Over a period of time, Figure 4.8(b) shows that due to the symmetric environment, most particles are centered on two possible positions to represent the



(a) (b)

(c)

Figure 4.8: The real robot experiments for single mobile robot localization in symmetric environments. (a) Initially, all the particles are spread over according to the uniform distribution. (b) Over a period of time, due to the symmetric environment, most particles are centered on two possible positions. (c) Finally, all the particles are tightly centered on the true position of the robot.

current belief of the robot. Finally, Figure 4.8(c) illustrates that all the particles are tightly centered on the true position of the robot, which indicates that the robot has successfully localized itself. The simulation tool shows that the above process takes 147.46 seconds, and the robot travels 19.26 meters. In the second run, the map has been changed to a smaller size, which is 183 cm $\times$ 122 cm. At this time the robot takes 123.76 seconds traveled 14.51 meters. The experimental results demonstrate that the single mobile robot needs large amounts of time to localize itself in a symmetric environment.

(b) Experiment of Multi-Robot Localization

In the second experiment iRobot Create and iRobot Discovery are used together within the same environment. Figure 4.9 shows our experimental environment. The

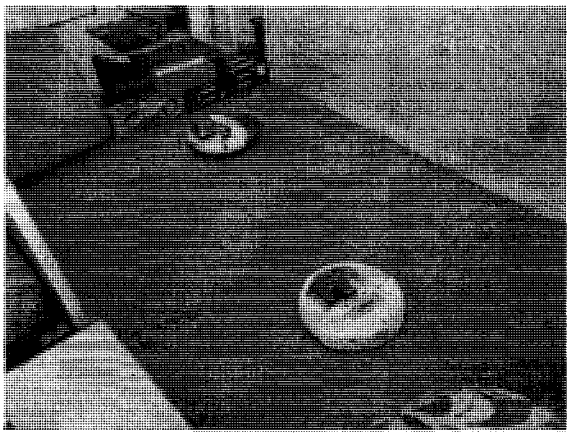


Figure 4.9: Experimental environment for multi-robot localization.

task of the robots is to perform global localization in the exactly the same maps as used in single mobile robot localization. However, a robot is able to exchange its belief during the detection with another robot. Our method uses 1000 particles to represent the belief for each robot. Figure 4.10(a) shows that the particle set in the left rectangular window represent the belief of iRobot Discovery, while the particle set in the right rectangular window represent the belief of iRobot Create. Initially, these two

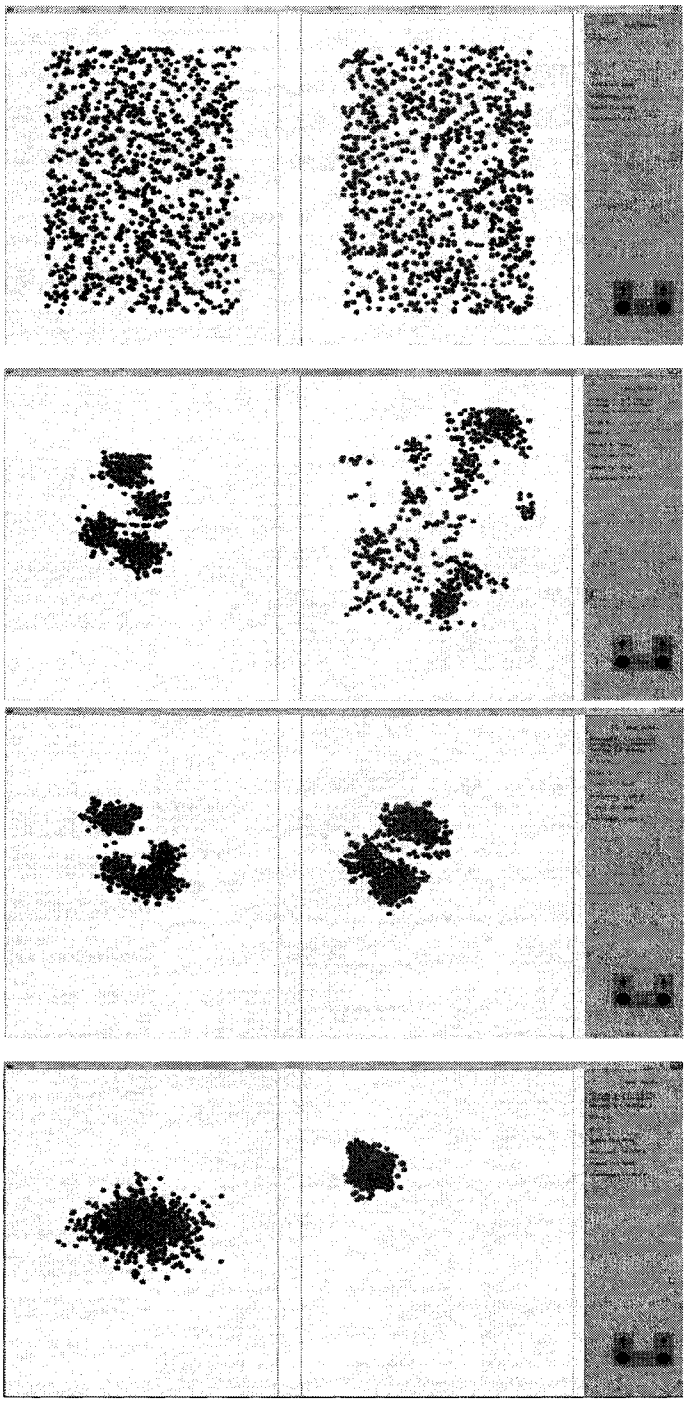


Figure 4.10: The real robot experiments for multi-robot localization in symmetric environments. (a) Initially, two particle sets are spread over according to the uniform distribution. (b) The belief of iRobot Discovery becomes more certain, while the belief of iRobot Create is still highly uncertain. (c) The iRobot Discovery uses its belief to refine the belief of iRobot Create. (d) Finally, both robots in our system have successfully localized themselves.

particle sets are spread over according to the uniform distribution, which indicates that two robot are global uncertainty to their environment. In Figure 4.10(b), because the iRobot Discovery has already obtained some knowledge about its surroundings by exploring the environment, most of the particles are centered on the true position of this robot to represent its current internal belief. Although iRobot Create has detected the wall, its belief is still highly uncertain at this time. According to our proposed approach the robot can benefit from the event of detection with another robot. Figure 4.10(c) shows a typical stage of this experiment where two robots detect each other. Both robots use our proposed method in chapter 3 to calculate entropies of their beliefs (triangle kernel function used for kernel density estimation), and then use these values to make a comparison. In this typical stage, the belief of iRobot Discovery has low entropy, which means this robot has more knowledge about its environment compared with iRobot Create. After this key event, the iRobot Discovery uses its belief to help the iRobot Create to refine its internal belief. Therefore, the localization process of the entire system is accelerated. Finally, Figure 4.10(d) shows that both robots in our system have successfully localized themselves by using our proposed approach.

Results: The experiment of multi-robot localization has been tested through two maps in different size. The maps' sizes are same as the previous experiment of single mobile robot localization. In the bigger map, the iRobot Create travels 10.84 meters using 102.77 seconds to localize itself, while the iRobot Discovery travels 12.72 meters using 110.83 seconds. In the smaller map, the iRobot Create travels 9.27 meters using 84.25 seconds to localize itself, while the iRobot Discovery travels 11.65 meters using 94.47 seconds. The experimental results demonstrate that compared to the experiment of single mobile robot localization, the event of robot detection and belief transfer mechanism proposed in our method not only reduces the distance traveled by each robot, but also notably reduces the time used for the process of localization for each robot. For example, without making use of information exchange, the iRobot Create needs 123.76 seconds in the smaller map, and 147.46 seconds in the

bigger map to determine its position. Under the help of iRobot Discovery, the localization time of iRobot Create in the symmetric environments is reduced by 30.31% to 102.77 seconds in the bigger map, and by 31.92% to 84.25 seconds in the smaller map. We summarize the experimental results for real robots shown in Table 4.1.

The experimental results for real robots.		Bigger Map (213.5 cm ×152.5 cm)			Smaller Map (183 cm × 122 cm)		
		Time (second)	Distance (m)	Number of detection	Time (second)	Distance (m)	Number of detection
Single-robot Localization (iRobot Create)		147.46	19.26	N/A	123.76	14.51	N/A
Multi-robot Localization	iRobot Discovery	110.83	12.72	4	94.47	11.65	8
	iRobot Create	102.77	10.84		84.25	9.27	
Localization performance of iRobot Create under the help of iRobot Discovery (The percentage of time saving)		30.31%			31.92%		

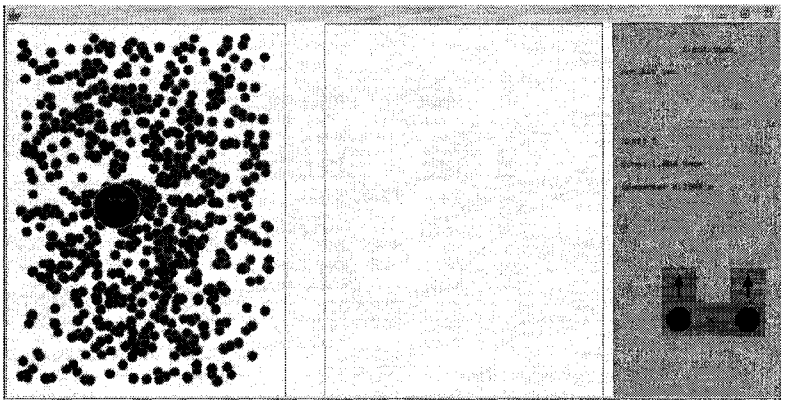
Table 4.1: The experimental results for real robots.

4.2.3 Simulation Experiments

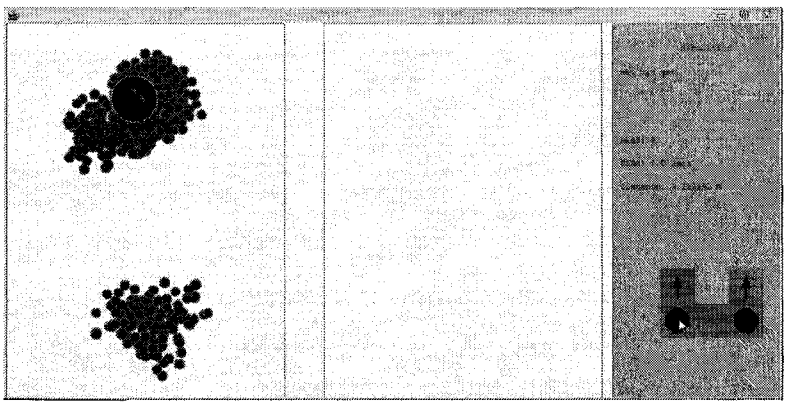
In this section, we present our simulation experiments. Robot detections were simulated by using the positions of the robots. Noise was added to robot motion according to the errors extracted by using our real robots.

(a) Experiment of Single Robot Localization in Symmetric Environment

In this simulation experiment we use a single robot. The task of this robot is to perform global localization in the rectangular free space shown in Figure 4.11. The



(a)



(b)

(c)

Figure 4.11: Simulation experiments of single mobile robot localization in symmetric environments. (a) Initially, all the particles are spread over according to the uniform distribution. (b) After sometime, due to the symmetric environment, most particles are centered on two possible positions. (c) Finally, all the particles are tightly centered on the true position of the robot, which indicates that the robot has successfully localized itself.

environment is particularly challenging for single robot system since its external environment is symmetry. We test this case using two maps in different size. In the first run, the map size is 366 pixels $\times$ 488 pixels. We use 1000 particles to represent the belief of each robot. Initially, all the particles are spread over according to the uniform distribution shown in Figure 4.11(a). Due to the symmetric environment, Figure 4.11(b) shows that most particles are centered on two possible positions to represent the current belief of this robot. Finally, Figure 4.11(c) illustrates that all the particles are tightly centered on the true position of the robot, which means that the robot has successfully localized itself. The simulation tool shows that the above process takes 109.45 seconds, and the robot travels 26.16 meters. In the second run, we change the map to bigger size 488 pixels $\times$ 610 pixels. This time the robot takes 236.26 seconds to travel 54.19 meters. The experimental results demonstrate that the single mobile robot needs a lot of time to localize itself in a symmetric environment without information exchange.

(b) Experiment of Multi-Robot Localization in Symmetric Environment

In this simulation experiment two robots, color in tint and color in dark, are used together shown in Figure 4.12. The task of the robots is to perform global localization in the exactly the same maps as used in the simulation experiment of single mobile robot localization. However, the robots are capable of exchanging their beliefs in this experiment. Our approach uses 1000 particles to represent the belief of each robot. The particle set in the left rectangular window represent the belief of the robot in dark, while the particle set in the right rectangular window represent the belief of the robot in tint. Initially, these two particle sets are spread over according to the uniform distribution, which means that two robots are global uncertainty about this environment shown in Figure 4.12(a). In Figure 4.12(b), because the robot in dark has obtained some knowledge about its surroundings by exploring the environment, most of the particles are centered on the true position of this robot to represent its current internal belief. Nevertheless, most of the particles for robot in tint are still spread

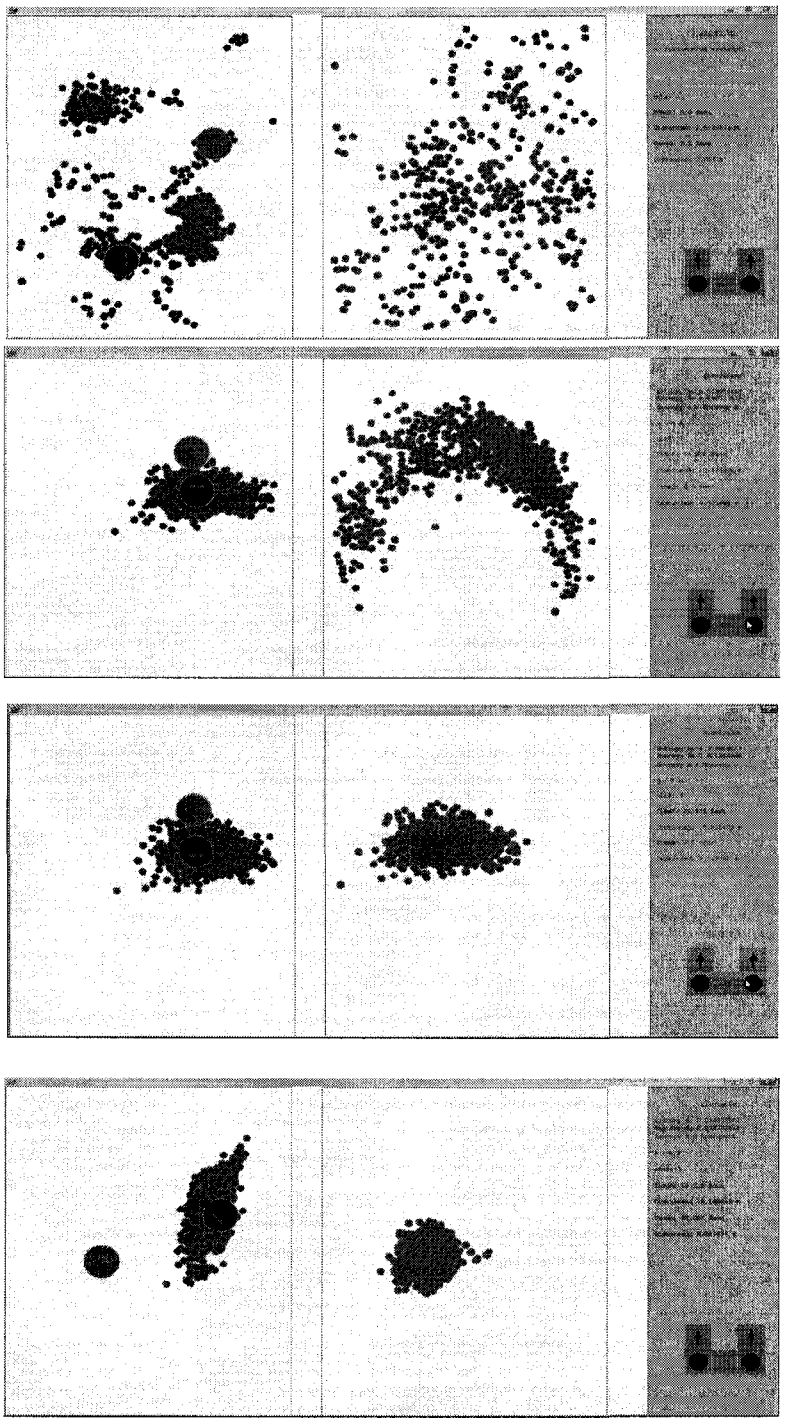


Figure 4.12: Simulation experiments for testing our approach in symmetric environments. (a) Initially, two particle sets are spread over according to the uniform distribution. (b) The belief of robot in dark becomes more certain, while the belief of robot in tint is still highly uncertain. (c) The robot in dark uses its belief to refine the belief of robot in tint. (d) Finally, both robots in our system have successfully localized themselves.

around the whole environment to reflect its weak knowledge about where it is in the given map. Figure 4.12(c) shows a typical stage of this simulation experiment where two robots detect each other. Both robots use the method proposed in chapter 3 to calculate entropies of their beliefs (triangle kernel function used for kernel density estimation), and then use these values to make a comparison. In this typical stage, the belief of robot in dark has low entropy, which indicates that this robot has more knowledge about its environment compared with the robot in tint. After this detection, the robot in dark uses its belief to help the robot in tint to refine its internal belief. Therefore, the localization process of the robot in tint is accelerated. Finally, Figure 4.12(d) shows that both robots in our system have successfully localized themselves through our proposed method.

Result: The experiment of multi-robot localization in symmetric environment has been tested through two maps in different size. The maps' sizes are same as the maps used in previous experiment of single mobile robot localization. In the smaller map, the robot in dark travels 6.45 meters using 29.75 seconds to localize itself, while the robot in tint travels 11.71 meters using 53.88 seconds. In the bigger map, the robot in dark travels 12.61 meters using 60.22 seconds to localize itself, while the robot in tint travels 8.38 meters using 30.09 seconds. The experimental results demonstrate that comparing to the simulation experiment of single mobile robot localization, the detection between robots and the belief transfer mechanism proposed in our approach not only reduces the distance traveled by each robot, but also notably reduces the time used for the process of localization for each robot. Without making use of detections, the robot in dark needs 109.45 seconds in the smaller map, and 236.26 seconds in the bigger map to uniquely determine its position. Our approach to multi-robot localization in symmetric environment reduces this time by 72.82% to 29.75 seconds in the smaller map, and by 74.51% to 60.22 seconds in the bigger map. Table 4.2 summarizes the simulation experimental results of multi-robot localization in symmetric environment.

The simulation experimental results in symmetric environment.		Bigger Map (488 pixel ×610 pixel)			Smaller Map (366 pixel × 488 pixel)		
		Time (second)	Distance (m)	Number of detection	Time (second)	Distance (m)	Number of detection
Single-robot Localization (robot in dark)		236.26	54.19	N/A	109.45	26.16	N/A
Multi-robot Localization	Robot in tint	39.09	8.38	3	53.88	11.71	5
	Robot in dark	60.22	12.61		29.75	6.45	
Localization performance of robot in dark under the help of robot in tint (The percentage of time saving)		74.51%			72.82%		

Table 4.2: The simulation experimental results in symmetric environment.

(c) Experiment of Multi-Robot Localization in Asymmetric Environment

In the experiment (b), we compare the performance of our multi-robot localization approach to the performance of single robot localization in the symmetric environments. In the following simulation experiment, we test our proposed approach in asymmetric environment which includes an obstacle represented by a small black rectangular box shown in Figure 4.13. Other than this obstacle, we have exactly the same simulation environment as described in the simulation experiment (b). Figure 4.13(a) shows that initially these two robots are both highly uncertain about their environment. Over a period of time, In Figure 4.13(b) robot in dark has obtained some knowledge about its surroundings by detecting the landmark. On the other hand, most of the particles for robot in tint are still spread around the whole environment. Figure 4.13(c) shows that after two robots detected each other, they both use our proposed method to calculate how much information is contained in their beliefs, and then use their entropies to make a comparison. In this typical stage, the robot in dark helps the robot in tint to refine its internal belief. Most of the particles for robot in tine are quickly centered on the robot's true position. By doing so, the localization process of

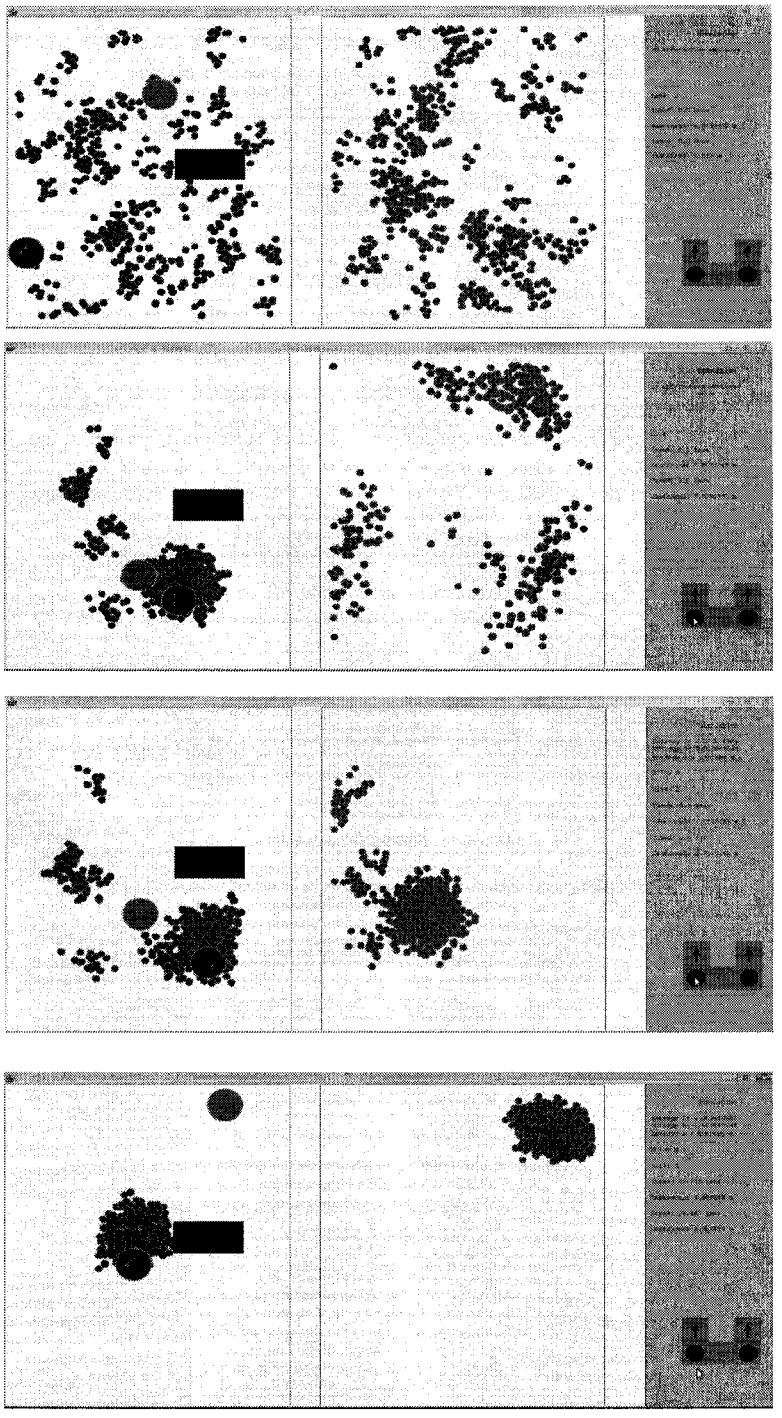


Figure 4.13: Simulation experiments for testing our approach in asymmetric environment. (a) Initially, these two robots are both highly uncertain about their environment. (b) The belief of robot in dark becomes more certain, while the belief of robot in tint is still highly uncertain. (c) The robot in dark uses its belief to refine the belief of robot in tint. (d) Finally, both robots in our system have successfully localized themselves.

robot in tint is speeded up. Finally, these two robots have successfully localized themselves shown in Figure 4.13(d).

Result: The experiment of multiply robots localization in asymmetric environment has been tested through two maps in different size. The maps' sizes are same as the previous experiment in (b). In the smaller map, robot in dark travels 5.13 meters using 23.67 seconds to localize itself, while robot in dark travels 6.73 meters using 36.70 seconds. In the bigger map, the robot in dark travels 6.50 meters using 28.51 seconds to localize itself, while robot in tint travels 7.33 meters using 38.14 seconds. The experimental results demonstrate that compared to the simulation experiment of single mobile robot localization, the robot detection and belief transfer mechanism proposed in our approach in the asymmetric environment not only reduces the distance traveled by each robot, but also notably reduces the time used for the process of localization. Without making use of detections, the robot in dark needs 109.45 seconds in the smaller map, and 236.26 seconds in the bigger map to uniquely determine its position. Our approach to multi-robot localization in asymmetric environment reduces this time by 78.37% to 23.67 seconds in the smaller map, and by 87.93% to 28.51 seconds in the bigger map. Because we have an obstacle in this simulation experiment, the symmetry of the environment breaks, which would help both robots within the environment to localize them easily and quickly. The experimental results also demonstrate that compared to the simulation experiment in (b), our approach to multi-robot localization in asymmetric environment reduces the localization time for robot in dark by 20.44% from 29.75 seconds to 23.67 seconds in the smaller map, and by 52.66% from 60.22 seconds to 28.51 seconds in the bigger map. Table 4.3 summarizes the simulation experimental results of multi-robot localization in asymmetric environment.

The simulation experimental results in asymmetric environment.		Bigger Map (488 pixel ×610 pixel)			Smaller Map (366 pixel × 488 pixel)		
		Time (second)	Distance (m)	Number of detection	Time (second)	Distance (m)	Number of detection
Single-robot Localization (robot in dark)		236.26	54.19	N/A	109.45	26.16	N/A
Multi-robot Localization	Robot in tint	38.14	7.33	3	36.70	6.73	5
	Robot in dark	28.51	6.50		23.67	5.13	
Localization performance of robot in dark under the help of robot in tint (The percentage of time saving)		87.93%			78.37%		
Localization performance of robot in dark compared with experimental results in symmetric environment (The percentage of time saving)		52.66%			20.44%		

Table 4.3: The simulation experimental results in asymmetric environment.

(d) Experiment of Multi-Robot Localization in More Complex Environment

In this simulation experiment, we test our proposed approach in more complex environment which includes two obstacles represented by two small black rectangular boxes shown in Figure 4.14. Other than this new obstacle, we have exactly the same simulation environment as described in the previous experiment (c). Figure 4.14(a) shows that initially both robots are highly uncertain. After some time, Figure 4.14(b) shows a typical stage that most of the particles of the robot in dark have already centered on the true poison of the robot, in the other hand the belief of the robot in tint is still very much uncertain. Figure 4.14(c) illustrates that after two robots perceived each other, they both use our proposed method to calculate how much information is contained in their beliefs, and then use these entropies to make a comparison. In this typical stage, the robot in dark helps the robot in tint to refine its internal belief. Most

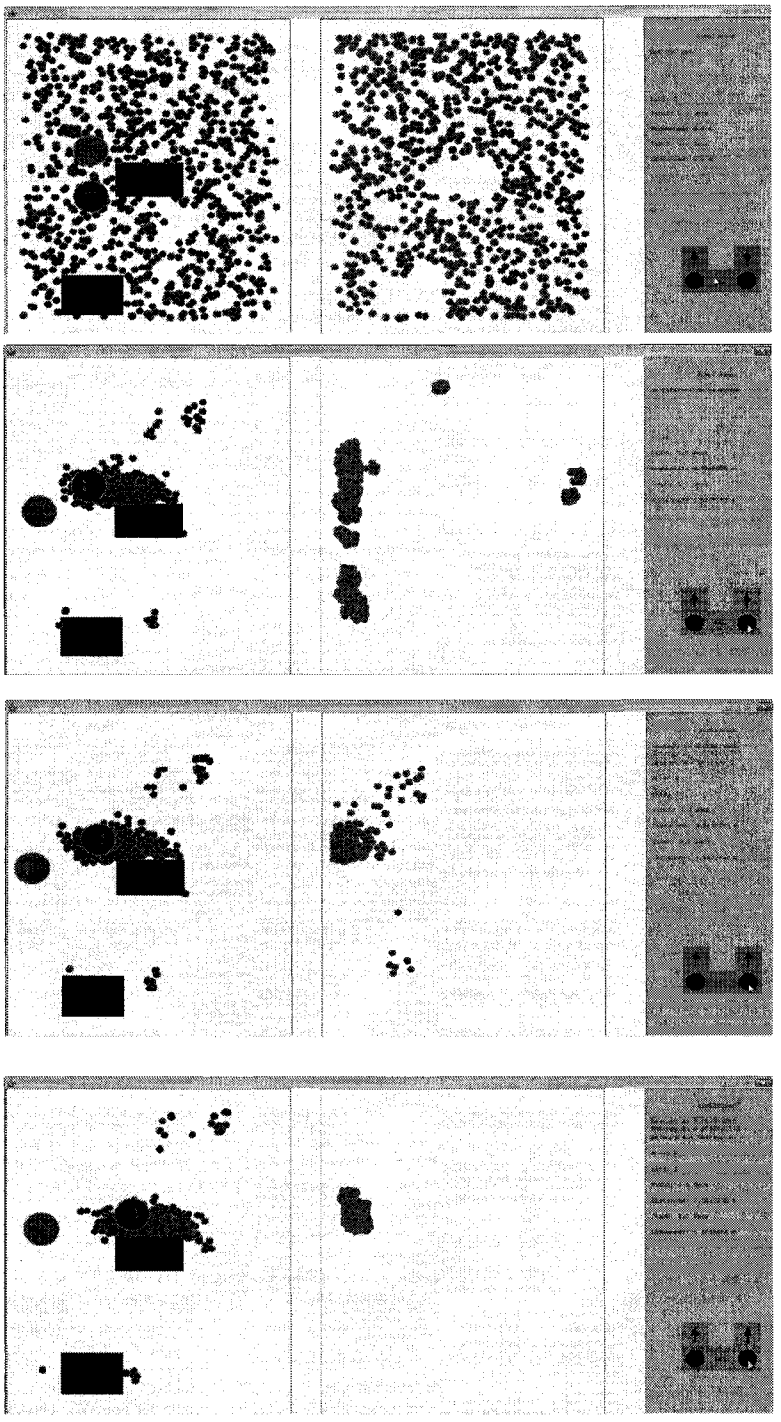


Figure 4.14: Simulation experiments for testing our approach in more complex environment. (a) Initially, these two robots are both highly uncertain about their environment. (b) The belief of robot in dark becomes more certain, while the belief of robot in tint is still highly uncertain. (c) The robot in dark uses its belief to refine the belief of robot in tint. (d) Finally, both robots in our system have successfully localized themselves.

of the particles for robot in tint are fast centered on the robot's true position. By doing so, the localization process of the whole system is accelerated. Finally, these two robots have successfully localized themselves shown in Figure 4.14(d).

Result: The experiment of multi-robot localization in more complex environment has been tested through two maps in different size. The maps' sizes are same as the previous experiment in (c). In the smaller map, the robot in dark travels 3.08 meters using 13.13 seconds to localize itself, while robot in tint travels 4.63 meters using 22.38 seconds. In the bigger map, the robot in dark travels 4.60 meters using 24.51 seconds to localize itself, while robot in tint travels 4.09 meters using 23.72 seconds. The experimental results demonstrate that compared to the simulation experiment single mobile robot localization, our proposed new mechanism of information exchange not only reduces the distance traveled by each robot, but also notably reduces the time used for the process of localization in the more complex environment. Without information exchange, robot in dark needs 109.45 seconds in the smaller map, and 236.26 seconds in the bigger map to uniquely determine its position. Our approach to multi-robot localization in more complex environment reduces this time by 88.00% to 13.13 seconds in the smaller map, and by 89.63% to 24.51 seconds in the bigger map. The experimental results also demonstrate that compared to the simulation experiment of multi-robot localization in symmetric environment, our approach to multi-robot localization in more complex environment reduces the localization time of robot in dark by 55.87% from 29.75 seconds to 13.13 seconds in the smaller map, and by 59.30% from 60.22 seconds to 24.51 seconds in the bigger map. Table 4.4 summarizes the simulation experimental results of multi-robot localization in more complex environment.

The simulation experimental results in more complex environment.		Bigger Map (488 pixel ×610 pixel)			Smaller Map (366 pixel × 488 pixel)		
		Time (second)	Distance (m)	Number of detection	Time (second)	Distance (m)	Number of detection
Single-robot Localization (robot in dark)		236.26	54.19	N/A	109.45	26.16	N/A
Multi-robot Localization	Robot in tint	23.72	4.09	3	22.38	4.63	4
	Robot in dark	24.51	4.60		13.13	3.08	
Localization performance of robot in dark under the help of robot in tint (The percentage of time saving)		89.63%			88.00%		
Localization performance of robot in dark compared with experimental results in symmetric environment (The percentage of time saving)		59.30%			55.87%		

Table 4.4: The simulation experimental results in more complex environment.

4.3 Summary

As described in section 4.2, experimental results, carried out in real and simulated environments, demonstrate that our approach can notably improve the localization performance, when compared to conventional single robot localization. Therefore, our proposed information exchange mechanism can yield significantly better localization results than single robot localization – at lower sensor costs, and relatively small communication overhead.

From the above experimental results one may notice that the simulation experimental results (percentage of time saving) are much better than the real robots experimental results. The differences between simulation experimental results and real

robots experimental results are possibly due to enormous uncertainties exist in our real robots experiments, which do not exist in the simulation experiments. There are several elements that contribute to these uncertainties. First, our real robots experimental environments are highly unpredictable (walls are constructed by moveable objects, e.g. tables, tool box, and microwave). Second, sensors are limited in what they can perceive. Limitations arise from several factors. The range and resolution of sensors of Roomba are subject to physical limitations. Sensors are also subject to noise, which disturbs sensor measurements in unpredictable ways and hence limits the information that can be extracted. For example, Roomba use its bumper to detect the external world, but the bumper sensors are not reliable. Third, robot actuation involves motors that are unpredictable (drift and slippage). Uncertainty is further created through the communication between the laptop and Roomba. Since the communication by using Bluetooth may cause the delay of data transmission, Roomba may lose the control command during the communication. Since the above four factors do not exist in our simulation experiment, there is no wonder that the simulation experimental results are much better than the real robots experimental results. Although the real robot experimental results are not good as the simulation experimental results, the overall performance of our proposed approach is still very good.

Chapter 5
Conclusion and Future Work

5.1 Conclusion

Cooperative multi-robot localization techniques use sensor measurements to estimate poses of robots relative to a given map of the environment. Existing approaches update a robot's pose instantly whenever it detects another robot. However, such instant update may not be always necessary and effective, since both robots' pose estimates could be highly uncertain at the time of the detection. In this thesis, we develop a new information exchange mechanism to collaborative multi-robot localization. We also propose a new scheme to calculate how much information is contained in a robot's belief by using entropy. Instead of updating beliefs whenever detection occurs, our approach first compares the beliefs of the robots which are involved in the detection, and then decide whether the information exchange is necessary. Therefore, it avoids unnecessary information exchange whenever one robot perceives another robot. On the other hand, this approach does allow information exchange between detecting robots and such information exchange always contributes positively to the localization process, hence, improving the effectiveness and efficiency of multi-robot localization.

Experimental results, carried out in real and simulated environments, demonstrate that our approach can notably improve the localization performance, when compare to the previous multi-robot localization at lower sensor costs, less computation costs and small communication overhead.

5.2 Future Work

The current approach can further be improved in the following aspects.

Determination of localization: One problem in our current system is that how to determine whether a robot has successfully localized itself. Presently, we consider that a robot has successfully localized itself, when all the particles tightly centered on the true pose of the robot. Currently we can only make such decision by our experience visually, because there is no established benchmark to help a system to determine if localization is success.

Active localization: The cooperative localization described in our approach is merely passive. The robots exchange their pose information locally during the detection time; however they do not change their actions so as to aid localization. The robot in our system is controlled through some random movement methods, and the robot's movement is not aimed at speeding up the localization process. Therefore, a desirable objective for future research is let the robot actively explores its environment based on the information gathered by itself or gathered from other robots so as to best localize themselves.

Identification of robots: Another limitation of the current approach arises from the fact that it must be able to identify individual robots. In our experiment, we only use two Roomba, and each Roomba equips with virtual wall sensor for perceiving another robot. If a robot's virtual wall sensor has reading, it knows exactly which robot is close to itself. However, on condition that our system has more than two robots, it would be hard for each robot in the system to identify which two robots are involving in the event of detection. In other words, if a robot's virtual wall sensor possesses reading, it is not possible to distinguish which robot being detected. In the future, such hardware limitation can be overcome by attaching the bar-codes to each

Roomba, and then use bard-codes reader to identify the robots which are involving the event of detection.

Preset time interval: In our approach, if two robots (Robot *A* and Robot *B*) have already exchanged their beliefs with each other, both robots' status variables of landmark detection will be set to false, which indicates that the robot (either Robot *A* or Robot *B*) can not exchange its belief with any other robot until the next time it detects a landmark. However, in situation in which one of the above robots detects another robot (e.g. Robot *C*) whose belief is highly uncertain after a short time since the last detection between Robot *A* and Robot *B*. The robot (either Robot *A* or Robot *B*) supposes to help to refine the belief of Robot *C* (since the belief of either robot *A* or Robot *B* is better than Robot *C*), but it is not allowed in our approach. If the robot (either Robot *A* or Robot *B*) is able to exchange information with robot *C*, the localization performance of the whole system may become better. One possible solution to the limitation is that we can preset a time interval. For instance, if the robot *A* has exchanged its belief with robot *B* and detects robot *C* within the preset time interval, robot *A* will be allowed to exchange information with robot *C*. Therefore, one objective for future research is to find out the appropriate preset time interval.

Comparison of beliefs: Finally, the robots exchange their belief whenever they fall into the third group which both robots have already detected the landmarks. In situations in which both robots' entropies of beliefs are almost same at the time of detection it might be more appropriate to delay the information exchange between two robots. This, however, requires that the method presets a threshold to determine when robots should exchange their beliefs. For example, if this threshold is less than the difference between the robots' entropies of beliefs, robots ought to exchange their beliefs; if the threshold is greater than the difference between the robots' entropies of beliefs, it might be more appropriate to delay the information exchange. Hence, a desirable objective for future research is to find out the suitable value of the threshold and to see of such threshold improves the performance.

Despite these limitations, our approach does provide an improved basis for information exchange during cooperative localization, and experiment results illustrate its effectiveness in practice. These results show that robots acting as a team are superior to robot acting individually.

Bibliography

[1] M. Betke and L. Gurvits. Mobile robot localization using landmarks. Technical Report SCR-94-TR-474, Siemens Corporate Research, Princeton, December 1993. will also appear in the *IEEE Transactions on Robotics and Automation*.

[2] R. Biswas and S. Thrun. A distributed approach to passive localization. *In Proceedings of the AAAI National Conference on Artificial Intelligence*, Pittsburgh, PA, 2005. AAAI.

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

[4] W. Burgard, A.B. Cremers, D. Fox, D. H¨ahnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1-2), 2000. To appear.

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

[6] W. Burgard, D. Fox, D. Hennig, and T. Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Menlo Park, August 1996. AAAI, AAAI Press/MIT Press.

[7] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun. Principles of Robotic Motion: Theory, Algorithms, and Implementation. MIT Press, Cambridge, MA, 2004.

[8] I.J. Cox and J.J. Leonard. Modeling a dynamic environment using a bayesian multiple hypothesis approach. *Artificial Intelligence*, 66:311–344, 1994.

[9] I.J. Cox and G.T. Wilfong, editors. *Autonomous Robot Vehicles*. Springer Verlag, 1990.

[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.

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

[12] D. Ferguson, A. Morris, D. Hähnel, C. Baker, Z. Omohundro, C. Reverte, S. Thayer, W. Whittaker, W. Whittaker, W. Burgard, and S. Thrun. An autonomous robotic system for mapping abandoned mines. *In S. Thrun, L. Saul, and B. Schölkopf, editors, Proceedings of Conference on Neural Information Processing Systems (NIPS)*, Cambridge, MA, 2003. MIT Press.

[13] 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, pp. 325–344, June 2000.

[14] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. *Proc. of the Sixteenth National Conference on Artificial Intelligence* (AAAI'99).

[15] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427, 1999.

[16] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427, 1999.

[17] T. Fukuda, S. Ito, N. Oota, F. Arai, Y. Abe, K. Tanake, and Y. Tanaka. Navigation system based on ceiling landmark recognition for autonomous mobile robot. In *Proc. Int'l Conference on Industrial Electronics Control and Instrumentation (IECON'93)*, volume 1, pages 1466 – 1471, 1993.

[18] Z. Ghahramani. Learning Dynamic Bayesian Networks. Lecture Notes In Computer Science, Vol. 1387, Pages: 168-197, 1997.

[19] J.-S. Gutmann and C. Schlegel. Amos: Comparison of scan matching approaches for self-localization in indoor environments. In *Proceedings of the 1st Euromicro Workshop on Advanced Mobile Robots*. IEEE Computer Society Press, 1996.

[20] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning.*

New York: Springer, 2001.

[21] J. Hertzberg and F. Kirchner. Landmark-based autonomous navigation in sewerage pipes. In *Proc. of the First Euromicro Workshop on Advanced Mobile Robots*, pages 68–73, 1996.

[22] P. Jensfelt and S. Kristensen. Active global localisation for a mobile robot using multiple hypothesis tracking. In *Proceedings of the IJCAI Workshop on Reasoning with Uncertainty in Robot Navigation*, pages 13–22, Stockholm, Sweden, 1999. IJCAI.

[23] L.P. Kaelbling, A.R. Cassandra, and J.A. Kurien. Acting under uncertainty: Discrete bayesian models for mobile-robot navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996.

[24] R. E. Kalman. A new approach to linear filtering and prediction problems. *Trans. ASME, Journal of Basic Engineering*, 82:35–45, 1960.

[25] F.H. Knight. Risk, Uncertainty, and Profit. Boston, MA: Hart, Schaffner & Marx; Houghton Mifflin Company, 1921.

[26] S. Koenig and R. Simmons. Passive distance learning for robot navigation. In L. Saitta, editor, *Proceedings of the Thirteenth International Conference on Machine Learning*, 1996.

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

[28] K. Konolige. Markov localization using correlation. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1999.

[29] R. Kurazume, S. Nagata, and S. Hirose, "Cooperative positioning with multiple robots," in *Proc. 1994 IEEE Int. Conf. Robotics and Automation,* vol. 2, Los Alamitos, CA, May 8–13, 1994, pp. 1250–1257.

[30] T. E. Kurt. 2006. Hacking Roomba. WILEY Press.

[31] J.J. Leonard, H.F. Durrant-Whyte, and I.J. Cox. Dynamic map building for an

autonomous mobile robot. *International Journal of Robotics Research*, 11(4):89–96, 1992.

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

[33] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.

[34] S. Mahadevan and N. Khaleeli. Robust mobile robot navigation using partiallyobservable semi-Markov decision processes. internal report, 1999.

[35] R. S., P. Norvig. Artificial Intelligence: A Mordern Approach. Englewood Cliffs, NJ: Prentice Hall, 2002.

[36] I. Nourbakhsh, R. Powers, and S. Birchfield. DERVISH an office-navigating robot. *AI Magazine*, 16(2):53–60, Summer 1995.

[37] Parzen E. (1962). On estimation of a probability density function and mode, Ann. Math. Stat. 33, pp. 1065-1076.

[38] I.M. Rekleitis, G. Dudek, and E.E. Milios. Multi-robot exploration of an unknown environment, efficiently reducing the odometry error. In M.E. Pollack, editor, *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, volume 2, pages 1340-1345, Nagoya, Japan, 23-29 Aug. 1997.

[39] W.D. Rencken. Concurrent localisation and map building for mobile robots using ultrasonic sensors. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2129–2197, Yokohama, Japan, July 1993.

[40] J. Reuter. Mobile robot self-localization using PDAB. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, CA, 2000. IEEE.

[41] S.I. Roumeliotis and G.A. Bekey. Bayesian estimation and kalman filtering: A unified framework for mobile robot localization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2985–2992, San

Francisco, CA, 2000. IEEE.

[42] S.I. Roumeliotis and G. A. Bekey. Distributed Multirobot Localization. *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 5, 2002, pp. 781 – 795.

[43] S. I. Roumeliotis and G. A. Bekey (2000). Distributed multi-robot localization. *IEEE Trans. Robotics and Automation*, 18(5):781-795.

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

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

[46] D.W. Scott. Multivariate Density Estimation. *Theory, Practice and Visualization*. New York: Wiley, 1992.

[47] D. W. Scott and Stephan R. Sain. Multi-dimensional Density Estimation. In C.R. Rao and E.J.Wegman, editors, Handbook of Statistics, Vol 23: *Data Mining and Computational Statistics*. Elsevier, Amsterdam, 2004.

[48] C.E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, vol. 27, pp. 379-423, 623-656, July, October, 1948

[49] B.W. Silverman. Density Estimation. London: Chapman and Hall, 1986.

[50] R. Simmons, R. Goodwin, K. Haigh, S. Koenig, and J. O'Sullivan. A layered architecture for office delivery robots. In *Proceedings of the First International Conference on Autonomous Agents*, Marina del Rey, CA, February 1997.

[51] A.F.M. Smith and A.E. Gelfand. Bayesian statistics without tears: a sampling resampling perspective. *American Statistician*, 46(2):84–88, 1992.

[52] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In I.J. Cox and G.T. Wilfong, editors, *Autonomous Robot Vehnicles*, pages 167–193. Springer-Verlag, 1990.

[53] J. R. Spletzer, and C. J. Taylor. A Bounded Uncertainty Approach to Multi-Robot Localization. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, USA , vol. 2, 2003, pp. 1258-1264.

[54] S. Thrun. Particle filters in robotics. *In Proceedings of the 17th Annual Conference on Uncertainty in AI (UAI)*, 2002.

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

[56] S. Thrun. Robotic mapping: A survey. *In G. Lakemeyer and B. Nebel, editors, Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[57] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141.

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

[59] S. Thrun, M. Bennewitz,W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. H¨ahnel, C. Rosenberg, J. Schulte, and D. Schulz. MINERVA: A second-generation museum tour-guide robot. In *Proc. of the International Conference on Robotics and Automation (ICRA '99)*, 1999.

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

[61] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Ne-fian, and P. Mahoney. Stanley, the robot that won the DARPA Grand Challenge. *Journal of Field Robotics, Forthcoming*.

[62] S. Thrun, J. Schulte, and C. Rosenberg. Interaction with mobile robots in public places. *IEEE Intelligent Systems*, pages 7-11, July/August 2000.

[63] N. Trawny, and T. Barfoot. Optimized Motion Strategies for Cooperative Localization of Mobile Robots. *Proceedings of the 2004 IEEE International*

Conference on Robotics & Automation, New Orleans, LA, April, 2004.

[64] B. Tribelhorn, and Z. Dodds. Envisioning the Roomba as AI Resource: A Classroom and Laboratory Evaluation.

[65] URL: http://store.irobot.com/home/index.jsp.

[66] URL: http://www.irobot.com/sp.cfm?pageid=289.

[67] URL: http://www.darpa.mil/grandchallenge05/.

[68] URL: http://www.darpa.mil/.

[69] URL: http://www.darpa.mil/grandchallenge/.

[70] URL: http://en.wikipedia.org/wiki/Stanley_%28vehicle%29.

[71] URL: http://en.wikipedia.org/wiki/Information_theory#Entropy.

[72] URL: http://en.wikipedia.org/wiki/Information_entropy.

[73] URL: http://en.wikipedia.org/wiki/Kernel_%28statistics%29.

[74] URL: http://en.wikipedia.org/wiki/Kernel_density_estimation.

[75] URL: http://www.quantlet.com/mdstat/scripts/spm/html/spmhtmlnode18.html.

[76] URL: http://www.irobot.com/.

[77] URL: http://www.irobot.com/sp.cfm?pageid=248.

[78]URL:http://todbot.com/blog/2006/02/14/roombacomm-roomba-control-communications-api-release-0/.

[79] URL: http://users.frii.com/jarvi/rxtx/.

[80] URL: http://www.processing.org/.

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

Appendix A --- iRobot Roomba Open Interface (ROI) Specification

iRobot® Roomba® Open Interface (ROI) Specification

Roomba Open Interface Commands Quick Reference

Command	Opcode	Data Byte 1	Data Byte 2	Data Byte 3	Data Byte 4	Etc.
Start	128					
Baud	129	Baud Code (0 – 11)				
Control	130					
Safe	131					
Full	132					
Power	133					
Spot	134					
Clean	135					
Max	136					
Drive	137	Velocity (-500 – 500)		Radius (-2000 – 2000)		
Motors	138	Motor Bits (0 - 7)				
Leds	139	Led Bits (0 - 63)	Power Color (0–255)	Power Intensity (0-255)		
Song	140	Song Number (0 – 15)	Song Length (0 – 15)	Note Number 1 (31-127)	Note Duration 1 (0-255)	Note Number 2, etc.
Play	141	Song Number (0 – 15)				
Sensors	142	Packet Code (0 – 3)				
Force-Seeking-Dock	143					

Baud data byte 1: Baud Code (0 – 9)

Baud code	Baud rate in bps
0	300
1	600
2	1200
3	2400
4	4800
5	9600
6	14400
7	19200
8	28800
9	38400
10	57600
11	115200

Motors data byte 1: Motor Bits

0 = off, 1 = on

Bit	7	6	5	4	3	2	1	0
Motor	n/a	n/a	n/a	n/a	n/a	Main Brush	Vacuum	Side Brush

Led data byte 1: Led Bits (0 – 63)

Dirt Detect uses a blue LED: 0 = off, 1 = on

Spot, Clean, and Max use green LEDs: 0 = off, 1 = on

Status uses a bicolor (red/green) LED: 00 = off, 01 = red, 10 = green, 11 = amber

Bit	7	6	5	4	3	2	1	0
LED	n/a	n/a	Status (2 bits)		Spot	Clean	Max	Dirt Detect

Power uses a bicolor (red/green) LED whose intensity and color can be controlled with 8-bit resolution.

Leds data byte 2: Power Color (0 – 255)

0 = green, 255 = red. Intermediate values are intermediate colors.

Leds data byte 3: Power Intensity (0 – 255)

0 = off, 255 = full intensity. Intermediate values are intermediate intensities.

Roomba Open Interface Sensors Quick Reference

Packet Code	Packet Size
0	26 bytes
1	10 bytes
2	6 bytes
3	10 bytes

Name	Groups	Bytes	Value Range	Units
Bumps Wheeldrops	0.1	1	0 – 31	
Wall	0.1	1	0 - 1	
Cliff Left	0.1	1	0 - 1	
Cliff Front Left	0.1	1	0 - 1	
Cliff Front Right	0.1	1	0 - 1	
Cliff Right	0.1	1	0 - 1	
Virtual Wall	0.1	1	0 - 1	
Motor Overcurrents	0.1	1	0 - 31	
Dirt Detector - Left	0.1	1	0 - 255	
Dirt Detector - Right	0.1	1	0 - 255	
Remote Opcode	0.2	1	0 - 255	
Buttons	0.2	1	0 - 15	
Distance	0.2	2*	-32768 – 32767	mm
Angle	0.2	2*	-32768 – 32767	mm
Charging State	0.3	1	0 - 5	
Voltage	0.3	2*	0 - 65535	mV
Current	0.3	2*	-32768 – 32767	mA
Temperature	0.3	1	-128 - 127	degrees C
Charge	0.3	2*	0 - 65535	mAh
Capacity	0.3	2*	0 - 65535	mAh

Bumps Wheeldrops

Bit	7	6	5	4	3	2	1	0
Sensor	n/a	n/a	n/a	Wheeldrop			Bump	Bump
				Caster	Left	Right	Left	Right

Motor Overcurrents

Bit	7	6	5	4	3	2	1	0
Motor	n/a	n/a	n/a	Drive Left	Drive Right	Main Brush	Vacuum	Side Brush

Buttons

Bit	7	6	5	4	3	2	1	0
Button	n/a	n/a	n/a	n/a	Power	Spot	Clean	Max

Charging State Codes

Code	Charging State
0	Not Charging
1	Charging Recovery
2	Charging
3	Trickle Charging
4	Waiting
5	Charging Error

VITA AUCTORIS

NAME	Huaicheng Su
PLACE OF BIRTH	Beijing, China
YEAR OF BIRTH	1980
EDUCATION	School of Electronic Engineering North China University of Technology Beijing, China 1999 – 2003 B.Eng.
	School of Computer Science University of Windsor Windsor, Ontario, Canada 2004 – 2006 B. Sc.
	School of Computer Science University of Windsor Windsor, Ontario, Canada 2006 – 2008 M. Sc.