Development of Collaborative SLAM Algorithm for Team of Robots

XU, WENBO

Development of Collaborative SLAM Algorithm for Team of Robots

Wenbo Xu

23th March 2014

School of Engineering and Computing Sciences
Durham University

A Thesis Submitted for the Degree of Master of Science by Research

Table of Content

List of Notation

s_t	pose of the robot at time *t*
θ_n	position of the *n*-th landmark
Θ	set of all *n* landmark positions
z_t	sensor observation at time *t*
Z^t	set of all observation $\{z_1, \ldots, z_t\}$
u_t	robot control at time *t*
u^t	set of all controls $\{u_1, \ldots, u_t\}$
n_t	data association of observation at time *t*
n^t	set of all data associations $\{n_1, \ldots, n_t\}$
$h(s_{t-1}, u_t)$	vehicle motion model
$g(s_t, \Theta, n_t)$	vehicle measurement model
P_t	control noise
R_t	measurement noise
$\overline{z_t}$	expected measurement of landmark
$z_t - \overline{z_t}$	measurement innovation
Z_t	innovation covariance matrix
S_t	FastSLAM particle set at time *t*
$S_t^{[m]}$	*m*-th FastSLAM particle set at time *t*
$\mu_{n_t,t}^{[m]}, \Sigma_{n_t,t}^{[m]}$	*n*-th landmark mean and covariance in the *m*-th particle
$N(\mu, \Sigma)$	normal distribution with mean μ and covariance Σ
$\omega_t^{[m]}$	importance weight of the *m*-th particle

Declaration

I certify that the work presented in this thesis is, to the best of my knowledge and belief original, except the material I cited in this thesis

I acknowledge that I have read and understood the University's rules, requirements, procedures and policy relating to my higher degree research award and to my thesis. I certify that I have complied with the rules, requirements, procedures and policy of the University.

Signature:_____ Date: _____

Acknowledgements

I would like to thank my thesis advisor, Peter Matthews, for supervising my work over this year. Without of his feedbacks on this thesis, I could not finish my study.

I would also like to thank my family, especially my parents for supporting and encouraging me to pursue this degree. Without their support, I would not have finished the degree.

Abstract

Simultaneous Localization and Mapping (SLAM) is a fundamental problem for building truly automatic robots. Varieties of methods and algorithms have been generated, and applied into mobile robots during the last thirty years. However, each algorithm has its strength and weakness. This thesis studies the most recent published techniques in the field of mobile robot SLAM. Specifically, it focuses on investigating robot path and landmark position estimating errors made by different methods. The Hybrid method, which uses FastSLAM method as front-end and uses EKF-SLAM method as back-end, combines both methods advantages, producing smaller errors on estimating robot pose. The Hybrid method solves the single robot SLAM problems by summing the weighted mean values of each particle in FastSLAM. The contributions of this thesis is it presents an alternate mapping algorithm that extends this single-robot Hybrid SLAM algorithm to a multi-robot SLAM algorithm. In this algorithm, each robot draws map of the environment separately, and robots could transfer their mapping information into a central computer. The central computer could merge the landmark positions from different robots. At last, a revised landmark position as well as its covariance will be calculated. Landmark positions are fused together according to two robots feature information by using Kalman Filters.

1 Introduction

The problem of Simultaneous Localization and Mapping (SLAM) has attracted many researchers in the field of robotics. SLAM addresses the problem of a mobile robot moving through an area without of any map information as a priori. The robot acquires observations of environment by its limited range-finder sensors and estimates its pose according to its odometry measurement. But both sensing observations and odometry measurement are affected by noise from either environments or robot systems. The aim of SLAM is to build a map of the environment and the path of the robot (Dellaert et al. 1999). This has been considered as a fundamental problem to build a truly autonomous robot system (Thrun 2002).

Robot SLAM could be considered as either single robot SLAM or multi-robot SLAM. If there is only one robot which is used for building maps of the environment, that is single robot SLAM problem. Similarly, if there are more than one robot working cooperatively to estimate maps of the environment. That is the multi-robot SLAM problem. Obviously, the control of group of autonomous robots could be more complicated compared to single robot in terms of system control, map fusing, memory requirement, but can bring more benefits to human beings, in terms of time consuming and map accuracy.

This thesis will focus on investigating SLAM algorithms for team robots. It will develop a multi-robot SLAM algorithm called multi-Hybrid SLAM in a simulation level, assuming data association is known. The data association problem in robot SLAM is another essential problem for mobile robot navigation, which is the process of relating features (landmarks), observed in the environment to features (landmarks) viewed previously or to features (landmarks) in a map. The existing single hybrid SLAM uses FastSLAM algorithm’s particles to present possible robot locations. For each particle, it contains all observed landmark location and covariance (uncertainty) information (Brooks & Bailey, 2009). Then, all these possible locations and positions

of landmarks are summed together by weighted mean, as well as their uncertainties (covariance matrix). Mapping of the environment and robot path are presented by the mean values and a covariance matrix.

It will extend the current hybrid SLAM algorithm to a multi-robot SLAM. The basic idea is that each robot could explore a certain area individually by using single Hybrid SLAM algorithm. Therefore, features of the environment are presented by a mean value and a covariance matrix. When two robots detect each other, they could transfer their feature information to each other according to their current location and the distance. When finishing the transfer, those features that have been detected by both robots could be summed together by Kalman Filter according to each feature's mean value and its covariance.

1.1 Motivation of SLAM

SLAM is a fundamental capability for mobile vehicle robots exploring in unknown environments where global position system (GPS) is not available. When exploring in these areas, robots should have some knowledge of their environment to estimate the path of navigation. However, to build a map of surrounding environment, it is necessary to know the true path of the robots as priori. This chicken-egg relationship between localization and mapping makes errors in robots' sensor readings enlarged by the errors in robots' motion (Thrun et al. 1998). Similarly, when robots move, their pose estimate is corrupted by measurement noise. To achieve the task of navigation in unknown environment, robots have to find appropriate estimation for both feature positions in the environment and robot locations. In robot SLAM, the process of estimating feature positions is called observation model. It gives robot locations and sensing information (distance and angles between robot and observed landmark) as input, the output is landmark position in the map. The process of estimating robot locations is called motion model. It gives robot pose for time *t-1* and the control as input. The output of current robot pose is calculated by adding the prose for time *t-1*

and current robot control.

The reasons for building multi-robot system is very different. However, one of the main motivations is the efficiency of multi-robots system. That is, compared to a single autonomous robot, a team of multiple robots can perform a mission better in terms of time cost and map quality. A team of robots could search the required environment cooperatively to directly reduce time cost of exploring; nevertheless, a team of robots usually have multiple points of view to the objectives in the environment. Environment objectives (e.g. landmarks) can be better estimated by fusing member robots' sensing data, which in turn increases the system effectiveness. Moreover, the reliability of multi-robot is higher than single robot because a team of robots could suffer one or two robots are damaged after robots begin their tasks. The rest of team members could finish those tasks that should be finished by those broken robots. Finally, instead of building a single powerful robot, building a team of robot can be easier and cheaper, can make the system tolerant to possible robots' faults, but can achieve complicated tasks as powerful as a single robot.

1.2 Application of SLAM

SLAM, as one of the most essential capability, has been widely used into autonomous mobile robots (Kuemmerle et al. 2009, Andreas et al. 2004, Cheein et al. 2010). Generally speaking, SLAM is used for autonomous vehicle navigating reliably cross those extreme areas where globally accurate position data (e.g. GPS) is not available, and are too distant, too dangerous, or too costly to allow human access into. These areas are, such as deep sea, underground, and on the surfaces of other planets. However, the applications of multi-SLAM may involve different fields, e.g., industrial robots, military and service robotics, and research and rescue robots, and they may be different in terms of missions, e.g., exploration and mapping, box pushing, military operation, unstructured environment navigation, and so on (Marjovi et al. 2009, Martijn & Andreas 2007, Wu et al. 2009).

One successful applications of SLAM in autonomous robots is used for assistant robots. Cheein (2010) applied Extended Kalman Filter (EKF) based SLAM into an assistance wheelchair to help those disabled people with navigation. The wheelchair is a semi-autonomous robot, which uses the biological signals to command the wheelchair. The operator (a C4 or C5 spinal cord injury) controls the direction of motion by means of electromyography (EMG) signals from the neck and the arm muscles. Then the wheelchair could map the environment and chooses paths for navigation instead of the patient.

SLAM is not only used for help with disabled people, but also applied with search and rescue program. In the NASA's disaster assistance and rescue team training (DART) project, researchers in the Stanford University robotics laboratory developed an air-ground cooperation SLAM algorithm into team robots. In this project, ground vehicles are used for rescuing. Aircrafts as they have a larger point of view, are used for searching and mapping the environments, and guiding land rovers to the areas with injuries.

1.3 SLAM Problem Definition

Considering a mobile robot moving cross an unknown environment, the robot could detect distance between robots and features in environment by using its sensors. But these sensor readings sometimes are not reliable. This is because, for example, laser and stereovision are sensitive to differences in lighting, and some feature surface does not reflect sound echoes well enough to be sensed by sonar. Robot controls are also suffered by noise. Simultaneous Localization and Mapping (SLAM) is the process of recovering the map of environment and the path of robot from the noisy controls and measurements.

If either the robot position or the map of environment is known with certainty, then the estimation of robot pose and feature of environment could use independent filters. However, for example, if the position of robot in the environment is unknown, errors

in robot's pose correlates errors in the map, and even worse, errors in robot's path could enlarge errors in map. Therefore, the state of robot pose and map must be estimated simultaneously.

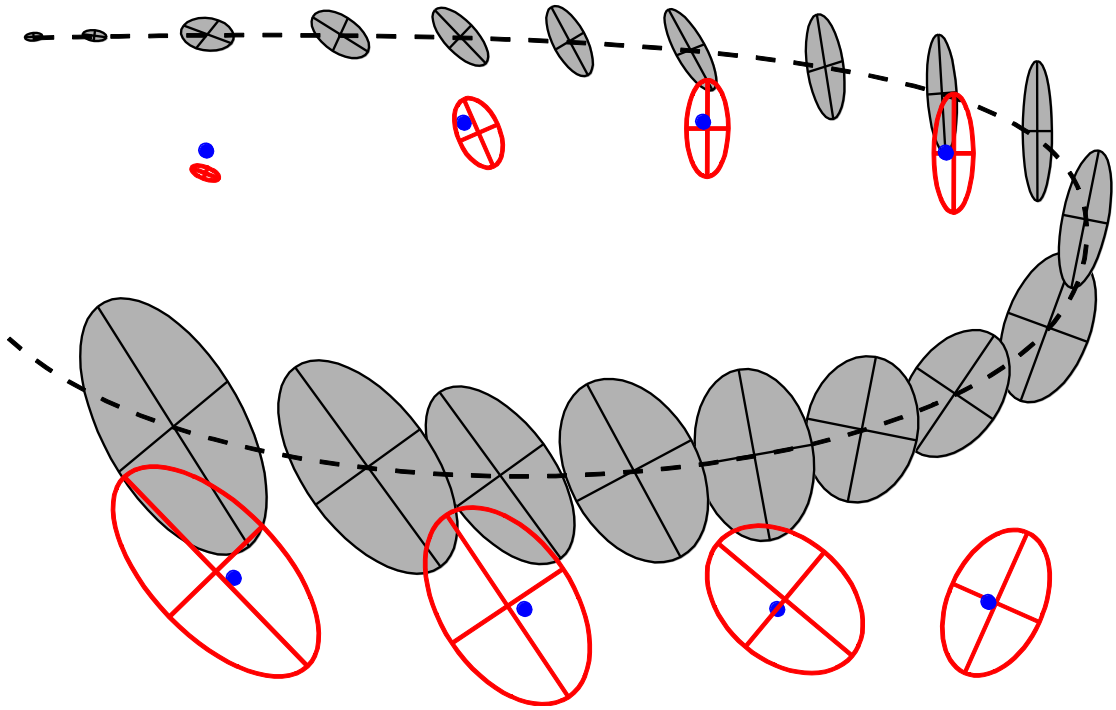


Figure 1: As the uncertainty of robot pose becomes larger, the uncertainty of nearby landmarks increases. The uncertainties of robot pose are drawn as shaded ellipses, and the uncertainties of landmarks are drawn as un-shaded ellipses (Montemerlo et al. 2002).

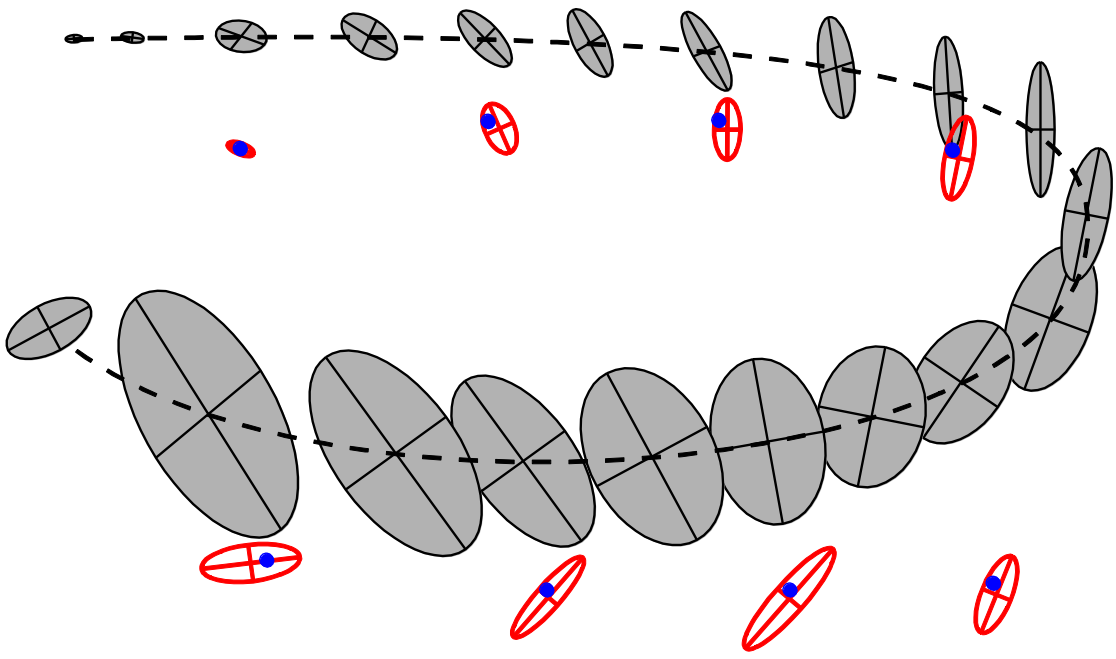


Figure 2: Revisiting known landmarks decreased not only the robot uncertainty, but also the uncertainty of landmarks previously observed (Montemerlo et al. 2002).

Figure 1.1 graphically shows the correlation in error between estimated robot pose and landmark positions. A robot moves from the left top along with the dashed line, observing nearby landmarks, drawn as circles. The shaded ellipses present the uncertainty of robot pose overt time. As a result of control error, the uncertainty in robot pose became larger and larger as robot moves. The estimations of nearby landmarks are drawn as un-shaded ellipses. It is clearly to see that as the uncertainty of robot pose increased, the uncertainty of newly observed landmarks increased as well.

In Figure 1.2, the robot finishes the loop, and detects a previously observed landmark. As the accuracy of first landmark is known with high accuracy, the uncertainty of robot pose will decrease. In turn, the uncertainty of other previously observed landmarks are decrease as well. Figure 1.2 clearly shows the correlated nature of SLAM problem. Errors in map are correlated errors in robot poses. Any observing information with high certainty will reduce uncertainty of previously observed landmarks.

1.4 Multiple Robot SLAM

For a multi-robot SLAM problem, this means that each robot holds its unique SLAM posterior according its current pose and a corresponding map of environment. When robots detect others and communicate their beliefs of the environment, such SLAM posterior can be combined, and narrow the hypothesis space for each robot.

1.4.1 Posterior Estimation

According to the previous description of the SLAM problem, a robot motion and observations of its environment are both noisy. Therefore, each control and observation can be thought as a mean value coupled with a probability uncertainty. This uncertainty is usually called covariance. For example, the mean value of robot

motion presents robot position in the environment, and the probability uncertainty shows how much we trust the robot stands in that area. On the other hand, the mean value of observation presents the relative distance between robot position and objective features in its environment, and the uncertainty expresses the errors in objective features. The idea to the SLAM problem is to estimate a posterior probability distribution over all possible maps and all possible robot poses, given the sensor readings and odometry information. This distribution is called the SLAM posterior.

This posterior estimation approach may seem incalculable at first. But by making certain assumptions of the state of the world, this posterior function can be computed efficiently and recursively. It will be discussed later in chapter 3.

1.5 Objectives

This MSc thesis aims to study the most recent published techniques in the field of Simultaneous Localization and Mapping (SLAM) and extend single robot SLAM algorithm into multi-robot SLAM. In particular, we will focus on SLAM techniques on 2D-SLAM with limited sensors and homogeneous multi-robot SLAM.

1. Conduct a literature review on SLAM method to obtain cutting edge techniques in this field.

2. Classify methods in the literature, to build up a data base.

3. Identify pros and cons of the main approaches based on readings.

4. Implement the some common methods to gain knowledge on its working principles.

5. Formulate possible improved methods for multi-robot SLAM.

1.6 Proposed method

This thesis will study the multi-robot SLAM problem by first analyzing popular single robot SLAM algorithms. A single robot SLAM is the fundamental problem for multi-robot SLAM. Because there are many successful single SLAM algorithms applied to multi-robot SLAM. Therefore, it must focus on analyzing the advantages and shortcomings of different SLAM algorithms. And then choosing an optimized algorithm and applying this algorithm on team robots. It found that the hybrid-SLAM algorithm has more advantages over others on estimating robot locations and landmark positions by doing comparing experiment on computers.

The basic idea for team robot SLAM is to use different robots' observation information to reduce estimation errors of robot pose and landmark positions. It is assumed that different team member robots could transfer their landmark information (positions and covariance) to the central computer. When the central computer receives landmark information from different robots, it can merge this information together by Kalman filter. For example, when two robots transfer the same landmark information by using two different landmarks' information, the computer could merge this information together by adding the variable on x-coordinate and the variable on y-coordinate according to Kalman filter, respectively. This is because the algorithm assumes variables on x-coordinate and y-coordinate are independent. They have no relationship between them.

For multi-hybrid SLAM algorithm, each member of robot will do the hybrid SLAM individually in the map. When they obtain a new landmark or update an old landmark, this new information will transferred to a global map. In the global map, if this new or updated landmark has been found by other robots, the algorithm could merge this landmark position with positions that are obtained by other robots.

The multi-robot SLAM algorithm will be done on a simulation level. Simulation will be finished on the Matlab. Robots in the simulation are assumed to equip with

limited laser sensors, which allow them to find landmarks or other robots. Large scale of mapping environment is not concerned because the experiment is done on a low personal computer. The population of landmarks is small.

1.7 Thesis Outline

1 **Chapter 2** will describe some relative works in the field of robotic SLAM problems. It first gives a problem definition of SLAM problem concerning to the uncertainty in robot motion and observation. The most important problem is that how to correctly estimate robot poses when robot controls and measurements are both noisy. Then, it shows some different literatures of robot SLAM techniques for both single and team robots. These techniques can be summarized as Kalman filters (KF) and Extended Kalman Filter (EKF), Expectation Maximization (EM), Sparse Extended Information Filter (SEIF), The Thin Junction Tree Filter (TJTF), Particle filter (PF) and FastSLAM, sub-map methods.

2 **Chapter 3** will formulate the SLAM problem and describe the theories and mathematics for EKF-based approach and PF-based approach. It will illustrate the basic theories on EKF-based SLAM method and PF-based SLAM method, and give a comparison for both methods in simulation level to show the estimation result in 2-D SLAM.

3 **Chapter 4** will describe the simplest version of the Hybrid approach SLAM algorithm with known data association. The basic idea for hybrid approach SLAM will be illustrated in this chapter. Experiment results will be compared with EKF SLAM algorithm and FastSLAM algorithm on the accuracy of robot pose and landmark positions.

4 **Chapter 5** will extend this Hybrid approach algorithm to multi-robot SLAM algorithm, and compare the result (the estimation errors on robot path and landmark positions) with a multi-FastSLAM algorithm.

5 **Chapter 6** will give the conclusions that in a simulation level, the hybrid method could be used for multi-robot SLAM and the result is better than multi-FastSLAM algorithm in accuracy. Future work will also be discussed in this chapter.

6 **Appendix** will show the Matlab code for multi-robot SLAM simulation. In this simulation, there are 18 landmarks in total in the environment, and two robots are given fixed path. Simulation set up refers to the data in Chapter 4.

2 Literature Review

This chapter will present an overview of the simultaneous localization and mapping problem (SLAM), along with the most common SLAM approaches from literatures. In the beginning, it first illustrates a short history of robot SLAM for readers to well understand researcher's primary ideas on reducing noise in robot localization and mapping problem, and how these ideas formulate to solve the problem of SLAM. Some different SLAM approaches for single robot will be discussed in section 2.2, and section 2.3 will discuss the most popular multi-SLAM techniques for team of robots. Then a short conclusion of this chapter will be proposed in section 2.4.

2.1 History of Robot SLAM

When mapping and localization were introduced by researchers in the early 1980's, the work at that time focused on solving mapping and localization independently. Robot mapping is the problem of acquiring an accurate map of the environment given some knowledge of robot's position and motion. The work in robotic mapping typically assumes that the robot's localization in the environment is 100% certain and focused mainly on analyzing the measurement data obtained from a noisy world to build up a map to present that environment. On the other hand, robot localization is another problem of estimating robot pose (robot's position and heading). Much work has been done on how to reduce the control error and robot slip problem when a robot is running through an area. In this situation, a map of working environment with landmarks is required as a prior knowledge for robot to determine where it is.

Smith et al. (1990) first introduced the idea of solving both of the mapping and localization problems, simultaneously. They developed a probabilistic method to indicate the spatial relationship between landmarks in an environment when robot is estimating its pose, currently. The map is represented as a set of landmark position and a covariance matrix is used to present the uncertainty of either the landmark or

the robot's pose. This kind of map today is usually mentioned as feature-based map, and this problem is called CML (Concurrent Mapping and Localization), but now more people would like to use SLAM (Simultaneous Localization and Mapping) (Thrun et al. 2005).

Since then, a probabilistic approach has become a standard way of solving the SLAM problem. Many issues associated with Kalman filter approach have been approved and other experiments by using an improved method, Particle filter technique, also accomplish the target successfully.

2.2 Current Single SLAM Solutions

Robotic mapping can be dated back to 30 years ago, and since 1990s probabilistic approaches have become dominant in robot SLAM, Kalman Filters (KF), Particle Filters (PF) and graphical SLAM became the three most popular solutions to SLAM (Sciliano & Khatib, 2007). The KF SLAM is the earliest solution to SLAM, which was fond by Smith et al. (1990). But this method has been a little unpopular because of its limitation of high computational complexity. PF is a new solution to solve the SLAM problem, and it provides some new solution to the data association problem. The third way of solving the SLAM problem is based on graphical properties, which has been successfully applied on some SLAM problem. In the followings there is a summary of classical and different solutions for current robot SLAM.

2.2.1 Extended Kalman Filter

Many of the original SLAM ideas came from a seminal paper submitted by Smith & Cheeseman (1986) and with their followers Csorba (1997), Guivant et al (2000), Huang & Dissanyake (2007), Liu & Thrun (2002), Baiyley et al. (2006), Tesli et al, (2011), who proposed the use of the Extended Kalman Filter (EKF) to estimate the SLAM posterior. The EKF represents the SLAM posterior as a high-dimensional, multivariate Gaussian by considering robots' odometry readings and sensing readings. Each multivariate Gaussian is parameterized by a mean value and a covariance matrix.

The mean value describes the most likely position of the robot and landmarks, while the covariance matrix describes the correlations between all pairs of state variables.

In the EKF SLAM, the complicated motion and the measurements are approximated by linear function. There are two substantial drawbacks in EKF-based SLAM: the quadratic computational complexity in the number of landmarks and the number of robots and the sensitivity to data association.

The disadvantage of the EKF to estimate SLAM posterior is computational complexity. There is a high demand for memory space for the EKF because the covariance matrix grows quadratically with the number of landmarks. Hence, the EKF-based SLAM algorithms could not work in large environments with the number of landmarks over several hundreds.

In a 2-D mapping world, the covariance matrix (a covariance matrix is a matrix that presents the robot path and landmark position estimation uncertainties in robot SLAM) is expressed as a $2N+3$ by $2N+3$ matrix, where N is the total number of landmarks in the map and 3 is the robot pose with x position, y position and headings in the map. Therefore, it is easy to prove that memory requirement for the covariance matrix grows quadratically (approximatelyN^2). Moreover, as the covariance matrix presents correlations between all features and other features in the map, any change in matrix will affect other elements. When new observations arrive, the EKF algorithm has to re-calculate all elements in the covariance matrix, which requires quadratic time. In practice, truly EKF-based algorithm is rarely used in real world applications. Instead, a large number of similar methods are used to reduce the computational complexity in the EKF updating process (Wan & Merwe, 2000).

The second problem with EKF-based SLAM approach is single-hypothesis data association. The data association problem is that each observation made by robots comes with a given mark that shows which landmark the robot is detecting. In the EKF-based SLAM approach, the robots are usually given known data association. But

in real world application, the associations between observations and landmarks are not clear. They must be determined by data association algorithms in order to well estimate robot pose and landmark positions.

The very basic idea for the EKF-based approach for data association is to assign each observation to landmarks by using Maximum Likelihood (ML) method. For each new observation of landmarks, it is assigned to the landmark which is the most likely to generate it. If the probability is lower than certain number, a new landmark is observed. The problem is that once a wrong data association between robot observations and landmarks is made, the EKF can not revise this mistake. If a large number of observations are mismatched, the EKF will give a bad estimation of posterior. It is possible to improve the accuracy of data association in the EKF, but it is expensive in computational cost.

2.2.2 Graphical based SLAM

The graphical based SLAM was first introduced in (Cheeseman & Smith, 1986). In graph based SLAM method, poses of robot and landmark positions are thought as nodes. Different nodes are connected by the lines. This lines are called constrains. Every consecutive pair of robot poses is constrained by a probability distribution conditioned to odometry measurement. Other constrains between robot poses and landmark positions are constrained by a probability distribution conditioned to feature observations. These constraints represent the log likelihood of the measurement and the motion model. The graphical based SLAM algorithm first interprets the sensor readings to extract the constraints, and then sum these constraints for the mimmum number, which could be thought as a least squares problem (Sciliano & Khatib, 2007).

If the observation noise is Gaussian and the data association is known, the goal of a graph based mapping algorithm is to compute a Gaussian approximation of the posterior over the robot trajectory from the initial time to current time. This involves computing the mean of this Gaussian as the configuration of the nodes that maximize the likelihood of the observation. Once this mean is known the information matrix of

the Gaussian can be obtained.

Graphical SLAM methods have the advantage that they scale to much higher-dimensional maps than EKF SLAM (Cheeseman & Smith, 1986). As the main problem of EKF SLAM is the quadratic covariance matrix. With the increasing number of landmarks, the computational complexity will increase quadratically. But there is no such problem in graphical based SLAM, as the update time of the graph is constant, and the amount of memory required is linear.

The disadvantage of graphical based SLAM is that it should optimize for the whole path of robot poses. If the robot path is long, the optimization may become cumbersome.

2.2.3 Expectation Maximization

The Expectation Maximization (EM) method is a stochastic approach developed by Dellaer et al. (2003), Thrun et al. (2004), Ruhnke et al. (2011). It was developed in the context of maximum likelihood (ML) estimation and it offers an optimal solution for map building. The EM is able to build a map when the robot's pose is known by means of expectation. There are two steps in EM SLAM: an expectation step (E-step), in which the posterior over robot poses is calculated for a given map, and maximization step (M-step), in which the most likely map is calculated given these poses expectations. The result is a series of increasingly accurate maps, while the initial map is an empty map (Thrun 2002).

The advantage of the EM method is that it solves the correspondence problem. It continues to localize the robot relative to the map generated in the E-step. The pose posteriors calculated in the E-step correspond to different hypotheses as to where the robot might have been. Therefore, it tried different correspondences. When building maps in the M-step, these correspondences are translated into features in the map, which, in turn, either get reinforced in the next E-step or disappear.

The problem of EM method is that when robot generates a new map to maximize the likelihood of the sensor readings in M-step, it should compute all the poses from the initial to current. It usually takes a long time to draw a map.

2.2.4 Sparse Extended Information Filter

Sparse Extended Information Filter (SEIF) is an optimal solution to the SLAM problem compared to EKF (Liu & Thrun 2002, Eustice & Ma 2005, Walter et al. 2007). The SEIF applies an alternative parameterization of KF, called information matrix. Instead of updating a covariance matrix in KF, the SEIF updates the information matrix (precision matrix). But the SEIF method uses the EKF to linearize the motion and measurement models. The advantage of SEIF method is that it makes the motion and measurement updates happen to be in constant time, if the covariance matrix is dense and the precision matrix may be sparse or many of its entries may be small.

2.2.5 Thin Junction Tree Filter

The Thin Junction Tree Filter (TJTF) is a SLAM algorithm based on the same principle of SEIF (Paskin 2003). It maintains a sparse network of probabilistic constraints between state variables, which enables efficient inference. The SLAM posterior is represented with a graphical model called Junction Tree. The size of this tree grows as new landmarks are incorporated to the map. TJTF has the advantage over SEIF that global maps can be extracted without any matrix inversion. This algorithm requires linear computation, which can be reduced to constant time with further approximation (Thrun 1998). But the disadvantages of TJTF method is that as the number of landmarks increase, the size of the tree grows. This size could be very large in sometime (Frank et al. 2004).

2.2.6 Particle Filter

The Particle Filter (PF) presents the posterior of SLAM estimation by a set of samples with different weights to present a proper path of the robot (Fox et al. 2000, Montemerlo et al. 2002). Each particle is attached with N independent landmarks

estimates (implemented as EKF), and hold a local map of the environment. The algorithm that is used for updating particle filter is called FastSLAM. The FastSLAM algorithm uses Rao-Blackwelized particle filter to solve the SLAM problem (Montemerlo et al. 2002, Stachiss & Burgard 2004, Baily, Nieto & Nebot 2006). In the algorithm, the uncertainty of robot motion is approximated with many hypotheses. Each hypothesis presents a possible robot path. Each path generates its own local map, in which landmarks are estimated by using Extended Kalman filters. This approach dramatically reduces the computational complexity compared to traditional EKF-based SLAM. As it contains multiple hypotheses to estimate the robot pose and data associations, the estimation of map could be more accuracy. The Particle method also has its problem on recording robot's trajectory.

One of the most important drawbacks for FastSLAM in use is it is difficult to record the path and its uncertainties. Robot pose is presented by choosing the particle with the highest weight. The weight value is a normalized character for all the particles. When a particle is selected to present current robot pose, the algorithm believes robot is absolutely there, and map features are read from this particle to represent the environment. The problem of this is the highest weighted particle may be changed at any time. So the robot path will not be continuous. But in the real world, a robot will continually move across an area.

This lack of record in remembering trajectory's uncertainty may also involve a problem that the estimations of SLAM posteriors are overconfident. FastSLAM or PF is discrete; sometimes the particle with the highest weights is not the peak value on real posterior of SLAM estimation. Therefore, it will make mistakes on estimating robot path and mapping the environment when applying on real robot SLAM.

2.2.7 Sub-map Method

The sub-map method is EKF-based approach for SLAM problem, which decompose global map into smaller sub-maps (Dissanayake et al. 2002, Roman & Singh 2005). A robot formulates a small sub-map with known robot pose. In this case, a robot is

usually assumed standing in the middle of the sub-map. When the robot moves out of the sub-map, it either creates a new sub-map or localizes itself in a previous formulated sub-map. A global map is a combination of these sub-maps in sparse network relationship. The sub-map method generates the same results as EKF-based SLAM, but with a lower computational requirement. Network Coupled Feature Maps, ATLAS, the Local Mapping algorithm, and Decoupled Mapping frameworks all consider relationships between a sparse network of sub-maps.

2.2.8 Hybrid Method

The Hybrid method is a combination SLAM method of EKF-SLAM and FastSLAM (Brooks & Bailey 2009). It contains both methods' strengths and avoids the weakness of these two methods. Map of the environment is produced by FastSLAM. Each particle's local map is then fused into an EKF-SLAM back-end. Then, the result in terms of robot poses and landmark positions are used as the initial information for the next time. The use of FastSLAM avoids linearization of the motion model and provides a high level of robustness to data association. The use of EKF-SLAM allows the uncertainty of robot path to be remembered, and avoids robot's poses become overconfident.

2.2.9 Summary of single robot SLAM

This thesis summarizes key properties of some of the most important algorithms in Table 2.1 and Table 2.2. The goal of this section is to clearly show the advantages and shortcomings of individual approaches.

In table 2.1, the map representation is summarized in the field representation, which has been defined before in details. The field sensor noise on the right of representation line means the kind of noise which the individual algorithm could process. Most algorithms only apply the Gaussian noise, but the EM algorithm could deal any noise. This means that the EM algorithm is more suitable for the real world environment. The field labeled uncertainty refers to the way uncertainty is represented in the resulting map. For the “Posterior poses and map”, results are presented by a robot

pose or landmark position coming with a belief. While for the “Maximum likelihood map”, the result is presented by only one map the best matching real word. Correspondence line indicates whether an algorithm can cope with unknown correspondence problem. The field of online means whether an algorithm is an online algorithm, which means the algorithm could produce the map before it finishing receiving all information data.

Table 2.1: lists of key properties of some of the most important algorithms

	Representation	Sensor noise	Uncertainty	Correspondence	Online
Kalman filter	Landmark position	Gaussian	Posterior poses and map	no	yes
Graphical map	Point obstacles	Gaussian	Maximum likelihood map	no	no
EM	Point obstacles	Any	Maximum likelihood map	yes	no
SEIF	Landmark position	Gaussian	Posterior poses and map	no	yes
TJTF	Landmark position	Gaussian	Posterior poses and map	no	yes
PF	Landmark position	Gaussian	Posterior poses and map	no	yes
Sub-map method	Landmark position	Gaussian	Posterior poses and map	no	yes
Hybrid method	Landmark position	Gaussian	Posterior poses and map	no	yes

Table 2.2 shows the advantages and disadvantages of filtering approaches applied into the SLAM. As it is disused before, the advantages and short comings is comparable. Any of these algorithms has its unique advantages and its short comings as well.

However, each of these algorithms has its limitation, but most of these filters have been successfully applied for single robot SLAM in real world.

Table 2.2: lists of advantages and disadvantages of filtering approaches applied into single robot SLAM.

Pros.	Cons.
Kalman Filter/ EKF	
-Handle uncertainty -High convergence	-Computational complexity -Poor in data association
Graph based SLAM	
-Linear memory requirement	-Map generates should compute the whole pass
Expectation Maximization (EM)	
-Optimal to map building -Solve data association	-Map generates should compute the whole pass
Sparse Extended Information Filter (SEIF)	
-Fast for high dimensional maps	-Poor in data association
The Thin Junction Tree Filter	
-Reduce computational complexity	-Large size of the tree
Particle Filter/FastSLAM	
-Low computational complexity	- SLAM posteriors are overconfident -Map accuracy very depend on the number of particles
Sub-map Method	
-Reduce memory usage -easy to build topological map for large environment	-Require multiple map merging -Poor in data association
Hybrid Method	
-Reduce memory requirement	-Poor in data association

Table 2.2 makes a survey of major algorithms in the field of robotic SLAM. The major paradigms in table 2.2 included Kalman filter method, graphical based method, EM method, sparse extended information filter method, the thin junction tree filter method, Particle filter method, sub-map method and hybrid method. Basic idea has been illustrated before, and their relative strengths and weaknesses have been pointed

out. In the following, there will be a short discussion of the algorithm.

It is noticed that all these algorithms described in the literature are of the state of the art method in robot SLAM, specifically in the indoor environment. These methods work with an assumption of robot navigating in a structured, static indoor environment. Robot odometry and sensing noise is assumed as a Gaussian white noise. However, the EM method could deal with non-Gaussian noise, but in the work (Corff et al. 2011) the authors set the noise as a Gaussian noise to compare the experiment results with EKF.

It is also noticed that there are three basic paradigms in robot SLAM algorithm. The first one is known as Kalman filter SLAM. It is the earliest solutions to robot SLAM problem. In this thesis, the EKF method, SEIF method, TJTF method, and sub-map method could be considered as the Kalman filter paradigms. As all these methods use Kalman filter to update robot pose and landmark position uncertainties to calculate the posterior. Representations of maps are presented by a mean value and the value's uncertainty. The limitations of these methods are to reduce the computational complexity of the quadratic covariance matrix. Although, the TJTF method and sub-map method reduce this problem to a linear complexity, they are still a Kalman filter based method to SLAM algorithm, which use a Taylor-series expansion to press the motion and observation model (the mean and uncertainties) from current time to the next time. Another limitation of the Kalman filter is they are all poor in dealing with the data association problem. Robot poses and map updates are very depending on correct data association.

The second paradigm of robot SLAM is graph based method. This method solves the SLAM problem by sparse optimization, which is to choose a robot pose to maximum matching the map with the highest probability or the minimum errors. Therefore, this family could be thought as a least squares problem. Graph based method and EM method could be considered as members of this family. The quadratic computation complexity in EKF paradigm has been well solved in this paradigm. From the

literatures, it could see that the EM method is not very much depend on data association, because in the M-step, robot pose which generated in the E-step are repeatedly correspond to different hypotheses (correct or incorrect) to find a best matching pose from all possible positions. After doing this, the incorrect hypotheses will be deleted as the incorrect data association would result in a small likelihood in the final result. The limitation of this method is it is an offline SLAM method. The offline SLAM means that the final map and robot pose should be calculated when all information data (odometry meter and sensing readings) are collected, or when robot produce the map at time *t*, it should calculate all information data from time *0* (the initial) to the current time *t*. With the increasing robot trajectory, robot will take a very long time to get the result.

The third paradigms of robot SLAM is based on particle filter. FastSLAM is a member of this family. In particle filter family, robot potential poses are presented by a set of particles. Each particle is given a weight to present the possibility where robot may stand on this position. Usually the particle with the highest weight is chosen as the robot current pose. And landmarks are presented by a mean and a covariance which is similar to the KF family. As FastSLAM uses particle filter to estimates robot poses, robot odometry error could be non-Gaussian, which means a robot drift and slip error could be set as real world situation. But for the observation model, this method can only produce the Gaussian noise sensor measurements. The advantage of this method is the low computational complexity, but the limitation is that map accuracy is much depends on the population of particles. An experiment shows that the more particles it chooses the accuracy the map will be. But with more particles added into this method, the calculation will increase. Another limitation of this method is the overconfident problem. For some reasons, most particles will locate into a certain area, but this area is far from the correct robot pose. When this happens, a map is formulated possibly wrong at that time, as robot pose only be chosen from the particles with highest weight.

This thesis will use the Hybrid method to solve the robot SLAM problem. The Hybrid method is a combination of EKF SLAM and FastSLAM. It uses the FastSLAM algorithm to estimate robot pose and uses the EKF to record robot trajectory and the map of environment. This is because it contains strengths and avoids weakness for both methods. It will first compare the experiment result with EKF and FastSLAM in terms of robot pose accuracy and landmark accuracy, and then it will expend this method to a two robot SLAM algorithm.

2.3 Successful Work on Multi-Robot SLAM

The above literatures are about SLAM methods for single robot. In recent years, researchers found that building a team of robots could have obvious advantages rather than single one in terms of stability, speed, and accuracy. Therefore, researchers' interests changed into team robot SLAM. In the followings, it will introduce some literatures on multi-robot SLAM.

2.3.1 Particle Filter Multi-SLAM

Howard (2006) works out an algorithm to solve multi-robot SLAM problem by using particle filter. This work can be considered as an extension work of (Fox et al. 2000). Robots could either know initial poses of all the robots or determine their relative poses (distance and headings) when they meet. This algorithm works under the assumption that robots can recognize each member in the team and determine their pose and transfer their motions and observations reliably. The basic idea is that a robot has no prior information of other robots, therefore robot performs single robot SLAM until it observes another one. When two robots (robot A and robot B) meet each other at time *t*, they determine their relative pose by using their own sensors. Then particle filter of robot A adds two additional parameters: a causal instance corresponding to forward motion of the robot, and an acausal instance corresponding to time-reversed motion of robot. Then the queued data of robot B (observation and odometry) is divided into causal instance and acausal instance as well. The causal instance is empty

at the time when they are meeting and it is updated with the odometry and observation data of robot B received by wireless. The causal queue is used to localize the observation of robot B into the map of robot A after the time when they meet. The acausal queue is used to combine the observation of robot B to the map of robot A before the time they meet. Integration of data in the acausal queue is considered to be a virtual robot moving backward until the beginning. So in this manner, one robot can combine the observation of all the robots into a single map. Therefore, each robot will produce a map of whole area after they meet each other. No map merging is required since all robots presented a map of whole environment.

2.3.2 SEIF Multi-SLAM

Thrun & Liu (2003) solved the problem of multi-robot SLAM by using Sparse Extended Information Filter (SEIF). The SEIF is an extended version of the information filter. Its estimation process is very similar to the EKF. The information matrix is the core in SEIF, which is like the covariance matrix in EKF. SEIF works by updating the information matrix. But there is a little different. SEIF only updates those observed landmarks in the information matrix. For those unobserved landmarks, SEIF keeps them the same as before. This filter has low estimation accuracy, but has high computation speed compared to EKF estimation.

Robots can obtain global map by building joint maps with their relative starting locations are unknown and landmarks are ambiguous under SEIF algorithm. Each robot perform individual SEIF based SLAM to formulate a partial/local map. Every local map is merged when all the searching is finished. To properly merge these local maps, the algorithm searches every landmark in local maps. For each identified landmark, the algorithm identifies three adjacent landmarks that fall into small radius. The relative distance and angles of each triangle are recorded as signs for map merging. The contribution of this algorithm is that it reduces the computation complexity of map merging by searching triangle signs instead of each grid of local maps.

2.3.3 EKF Multi-SLAM

Dissanayake et al. (2000) and Seib et al. (2011) proposed a multi vehicle map building algorithm based on EKF for multiple vehicles with knowing initial poses and unique landmarks. The main contribution of this paper is the authors produced an effective communication among team members to update their observations and making the system reliable.

Zhou & Roumeliotis (2006) proposed an algorithm by using EKF to estimate robot poses and landmark positions. The basic idea is that robots are arranged or randomly meet at least once for each two robots. Robots' local map can be transferred to other robots when they meet based on sensing measurement. The problem is that when they transferring local map data, there may be duplicate landmarks due to the uncertainty of landmark estimation. The authors developed Sequential Nearest Neighbor Test to detect and combine duplicate landmarks. The idea is that if L1 and L2 present the same landmark in different local map, the distance between them in merged map should be zero or within a small number. Therefore, the authors set a small valid number as a threshold to determine if the two landmarks are the same one. If the distance is smaller than the threshold number, then they are considered as the same one in the merged map.

2.3.4 Other Methods

Wang et al. (2007) presented a solution to multi-robot SLAM problem based on D-SLAM (Decoupled SLAM) framework. Robots start without known relative locations. Each robot builds local maps of its own by traditional EKF SLAM algorithm. D-SLAM framework is used to merge those partial maps. Since robots apply EKF SLAM algorithm independently to each other, the local maps are uncorrelated with each other as well. The map merging is conducted by an algorithm developed from image registration method. The idea of image registration is to find the transformation of different local maps to join them together. The transformation T can be calculated by correlating features in local maps. At last, a correspondence

value is proposed to verify feature correspondence. Transformation with large correspondence value presents maps with high similarity.

Change et al. (2007) proposed a novel algorithm to build a Topological/Metric maps by multi-robots. In this paper, each robot performs a Rao-Blackwellized particle filter to formulate a metric map (vertex). The authors set each metric map (vertex) with a fixed set. The initial location of the robot is set to the center of the square. When a robot moves into a new metric map or when two robots meet each other, an edge is build to present the difference of coordinate in the two metric. Thereby, it is very easy to solve the map merge problem in multi-robot SLAM.

2.3.5 Summary of multi-robot SLAM

From the previous literatures, it could know that the simplest way of solving multi-robot SLAM is to apply the map merging algorithms. Different robots' local map are fused together to formulate a global map. The difficulty of this idea is how to correctly find those overlapped area which are made by different robots. It is also known that EKF, PF, and SEIF SLAM methods have been applied into multi-robot SLAM. The idea of these methods to estimate robot poses considering the information collected by all team members.

This thesis will follow the same idea to expand the Hybrid SLAM method into two robots SLAM. As it discussed in Chapter 1, the difficulty of this method is how to use the additional information from the detected robot to narrow robot's SLAM posterior. In the Multi-Hybrid SLAM algorithm, the positions of landmarks will be summed together by using Kalman filter. The landmark posterior of the multi-hybrid SLAM is the sum of each robot's posterior for landmarks.

2.4 Conclusion

It is clear that researchers became more interested in developing algorithms for team robot rather than single robot. This is because a team of robots have obvious

advantages rather than single robot to accomplish a task in terms of time consuming, stability and energy cost. Therefore, it is necessary to work on team robot SLAM algorithm. It is notable that most literature focuses on solving computational complexity, but few papers work on increasing estimation accuracy.

There are two main ideas for generating multi-robot SLAM algorithms. The first one is directly applying single robot SLAM algorithms into each team robot to map the environment independently, and then propose a method to merge these local maps into a global map. The difficulty of this idea is how to correctly and efficiently find out the overlapped area between different local maps for map merging. The second idea is expanding current single robot SLAM algorithm for team robots. The algorithms, such as KF, SIEF, FastSLAM and sub-map method have been successfully applied into multi-robot SLAM algorithms already.

In this thesis, it will apply the single robot SLAM algorithm into multi-robot SLAM. This is because, with multiple points of view, team robots could have a better knowledge in estimating their poses and the environment. And also, robots could have a better decision on navigation by sharing their information.

2.4.1 Proposed Solution

The multi-Hybrid SLAM method will be proposed in this thesis. The multi-Hybrid SLAM method is formulated by expanding single Hybrid SLAM algorithm. It assumes that there are two robots exploring a certain area, knowing each other's location at initially. Each robot performs a single Hybrid SLAM algorithm when they are moving. The resulting mapping information of each individual robot will be transferred to the central computer to merge different robots' map into more accurate map by using Kalman filter which will be disused in Chapter 4.

3 SLAM

The previous chapters illustrated background information on robot simultaneous localization and mapping problem, as well as some typical solutions to robot SLAM problem. In this chapter, methodologies and mathematical details on EKF and FastSLAM will be presented. This chapter will illustrate how these two methods filter out noise on robot controls and sensor readings respectively when a robot moves across an unknown area.

3.1 SLAM Posterior

The pose of the robot at time *t* will be denoted s_t. For robots operating in a planar environment, this pose consists of the robot's *x*-*y* position in the plane and its heading direction. The complete trajectory of the robot, consisting of the robot's pose at each time step, will be written as s^t,

$$\mathrm{s}^{\mathrm{t}} = \{\mathrm{s}_1, \mathrm{s}_2, \ldots\ldots \mathrm{s}_{\mathrm{t}}\}.$$

The environment of the robot's working area can be presented as a set of point landmarks. These landmarks present the locations of features extracted from sensor data, and will be denoted $\{\Theta_1, \Theta_2 \ldots . \Theta_N\}$. The entire map of the environment will be written as Θ.

As the robot moves through the environment, it will collect relative information about its own motion. This information can be measured by using odometer attached on the wheels of the robot, or by recording the control command executed by robot. In robot SLAM problem, any measurement of the robot's motion will be referred as a control. The control at time *t* will be written as μ_t. Similarly, the set of all controls executed by the robot will be written as μ^t. Therefore,

$$\mu^t = \{\, \mu_1, \, \mu_2, \ldots\ldots \, \mu_t\}.$$

As the robot moves through the environment, it will observe nearby features (landmarks). In the most common cases of the SLAM problem, the observation information will contain both the range and bearing to nearby obstacles. Hence, the observation at time t can be written as z_t. The set of all observations collected by the robot will be written as z^t,

$$z^t = \{ z_1, z_2, \ldots\ldots\ z_t \}.$$

In robot SLAM, sensor measurements can be decomposed into information about individual landmarks. Each observation of landmark can be incorporated independently from the other measurements. Thus, it is assumed that each observation provides information about the location of one landmark relative to the robot's current pose. The variable *n* represents the identity of the landmark being observed. In practice, the identities of landmarks usually can not be observed, because the robot has limited knowledge to identify landmarks. The identity of the landmark corresponding to the observation will be written as n_t, where $n_t \in \{1,2, \ldots . N\}$. For example, $n_5 = 4$ means that at time *t=5*, the robot observed landmark number *4*. Landmark identities are commonly referred as data associations or correspondences. The set of all data associations will be written as n^t,

$$n^t = \{ n_1, n_2, \ldots\ldots\ n_t \}.$$

For simplicity, it is assumed that at time *t*, the robot observed exactly one measurement z_t and executes exactly one control μ_t. The goal of SLAM is to recover the best estimation of the robot pose s^t and the map Θ, given the set of noisy observations z^t and controls μ^t. In Bayesian probabilities, it is expressed by the following posterior, also referred as the SLAM posterior, which has been mentioned before in Chapter 1 and Chapter 2,

$$p(s_t, \Theta | \, z^t, \, \mu^t, \, n^t).$$

3.2 Mathematical Derivation

It is not difficult to understand that the current pose of the robot s_t can be written as a probabilistic function of the pose at the previous time step s_{t-1} add the control μ_t executed by the robot. This function is referred as the motion model. It describes, firstly, how controls change the pose of robot, and secondly, how the control noise influences the uncertainty of robot pose estimation. The motion model is written as,

$$p(s_t | s_{t-1}, \mu_t).$$

Sensor readings collected by the robot are also presented by a probabilistic function. This function is referred as the measurement model. The observation z_t is a function of the landmark being observed and the pose of s_t. The measurement model describes the error model of the robot's sensor. The measurement model is written as,

$$p(z_t | s_t, \Theta_t, n_t).$$

Therefore, the SLAM posterior at time *t* can be computed recursively by using the motion model, the measurement model and the SLAM posterior at time *t*-1. This recursive update rule is called Bayes filter for SLAM, which is the basis for the majority of online SLAM algorithm.

3.2.1 Bayes Filter Derivation

The Bayes Filter can be derived from the SLAM posterior. Assume that at time *t*, a robot moves across an area without prior knowledge about this environment. It obtains information of features by using its sensors readings and estimates its pose by using odometer readings. Therefore, the posterior of SLAM can be written as,

$$p(s_t, \Theta | z^t, \mu^t, n^t) = \eta\, p(z_t | s_t, \Theta, z^{t-1}, \mu^{t-1}, n^t) p(s_t | z^{t-1}, \mu^t, n^t).) \quad (3.1)$$

Where η is a normalizing constant number. According to the Markov chain assumption, we know that the observation factor z_t is isolated from the previous

observations and controls. It is only relevant to the current robot pose s_t, the map Θ, and the latest data association n^t. Hence the posterior can be written as,

$$p(s_t, \Theta | z^t, \mu^t, n^t) = \eta\, p(z_t | s_t, \Theta, n_t) p\left(s_t, \Theta \middle| z^{t-1}, \mu^t, n^t\right). \quad (3.2)$$

Then, according to the law of Total Probability theorem, we can expand the rightmost term on the pose of the robot at time *t*-1,

$$p(s_t, \Theta | z^t, \mu^t, n^t) =$$
$$\eta\, p(z_t | s_t, \Theta, n_t) \int p\left(s_t, \Theta \middle| s_{t-1}, z^{t-1}, \mu^t, n^t\right) p(s_{t-1} | z^{t-1}, \mu^t, n^t) ds_{t-1}. \quad (3.3)$$

Then, we can expand the leftmost term inside the integral by using the definition of conditional probability. This is because in robot SLAM problem, the map of environment is independ robot pose at any time. The true map of environment is static, and it would not change. Therefore, we get,

$$p(s_t, \Theta | z^t, \mu^t, n^t) =$$
$$\eta\, p(z_t | s_t, \Theta, n_t) \int p(\Theta | s_{t-1}, z^{t-1}, \mu^t, n^t) p(s_t | s_{t-1}, z^{t-1}, \mu^t, n^t) p(s_{t-1} | z^{t-1}, \mu^t, n^t) ds_{t-1}.$$
(3.4)

We can note that, robot pose s_t in the second term inside the integral is only a function of previous robot pose s_{t-1} and latest control μ_t as it is previously described as the motion model,

$$p(s_t, \Theta | z^t, \mu^t, n^t) =$$
$$\eta\, p(z_t | s_t, \Theta, n_t) \int p(\Theta | s_{t-1}, z^{t-1}, \mu^t, n^t) p(s_t | s_{t-1}, \mu_t) p(s_{t-1} | z^{t-1}, \mu^t, n^t) ds_{t-1}.$$
(3.5)

Then, we can combine the first term and the third term in the integral,

$$p(s_t, \Theta | z^t, \mu^t, n^t) =$$
$$\eta\, p(z_t | s_t, \Theta, n_t) \int p(s_t | s_{t-1}, \mu_t) p(s_{t-1}, \Theta | z^{t-1}, \mu^t, n^t) ds_{t-1}. \quad (3.6)$$

To simplify this equation, since the current control and data association n_t provides no new information about s_{t-1}, we can delete these factors in the rightmost term of the integral. Then, it is easy to see that the result is a recursive formula to compute the SLAM posterior at time *t* by providing the SLAM posterior at time *t*-1. And computation is based on calculating the motion model $\int p(s_t|s_{t-1},\mu_t)ds_{t-1}$, and the measurement model $p(\ z_t|s_t,\Theta,\ n_t)$,

$$p(\ s_t,\Theta|\ z^t,\ \mu^t,\ n^t) =$$
$$\eta\ p(\ z_t|s_t,\Theta,\ n_t)\int p(s_t|s_{t-1},\mu_t)p(s_{t-1},\Theta|\ z^{t-1},\ \mu^{t-1},\ n^{t-1})ds_{t-1}. \quad (3.7)$$

3.3 Kalman Filtering

Most ideas of SLAM algorithms came from a seminal paper by Smith & Cheeseman (1986). They proposed the application of the Extended Kalman Filter (EKF) to estimate the SLAM posterior.

The EKF presents the SLAM posterior as a multivariate Gaussian parameterized by a mean μ_t and a covariance matrix Σ_t. The mean values describe the most likely position of the robot and landmarks, and the covariance matrix encodes the pairwise correlations between all pairs of other state variables.

To keep this problem as simple as possible, here it is only consider a 1-dimensional example of EKF estimation robot pose. Suppose a mobile robot is moving through a line on the ground. It makes two different observations z_1 and z_2 with two different sensors. It is known that each sensor has its own accuracy. Therefore, the observation detected by the first sensor has an error modeled by a Gaussian with standard deviation σ_1. The error of the second sensor is also normally distributed with standard deviation σ_2. The robot would like to combine both readings into a single estimation to best estimate the distance between its position and the landmark.

To our common sense, if both sensors have the same error ($\sigma_1 = \sigma_2$), it is just take

the average of both values. If the first sensor, for example, is greatly superior $\sigma_1 \ll \sigma_2$, the robot will keep z_1 as the estimation, and vice versa. In any other case, the robot would like to formulate a weighted average of both readings to compute an estimation of *z*. Therefore, the question is which the best weighted average is.

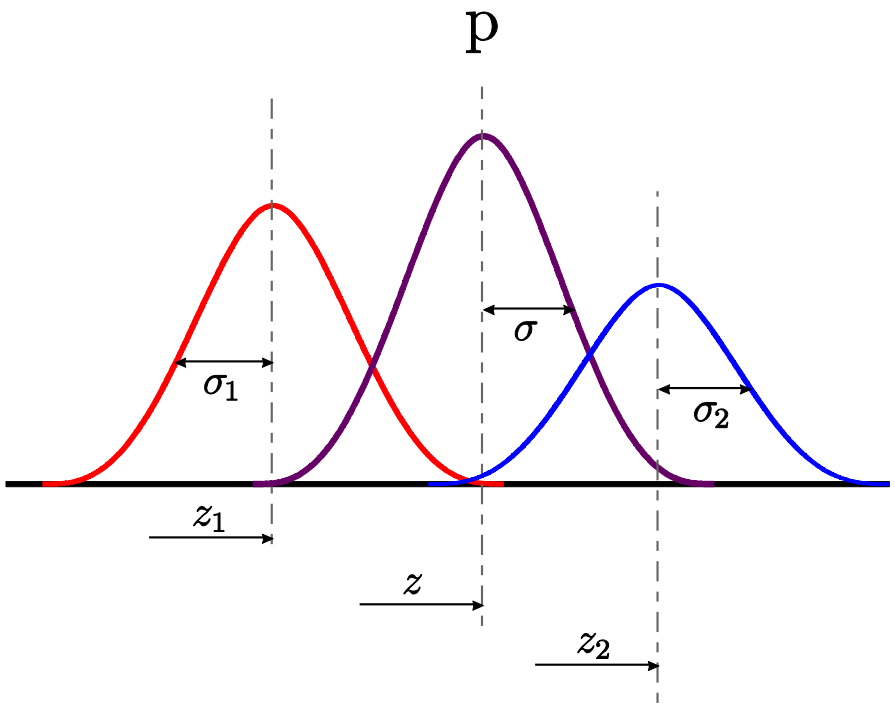


Figure 3.1: Combination of two different normal distributions by using Kalman Filter

According to the Kalman Filter, we know that one possibility is weighting each reading inversely proportional to its precision, that is,

$$z = \frac{\frac{z_1}{\sigma_1^2} + \frac{z_2}{\sigma_2^2}}{\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}}, \tag{3.8}$$

Or simplifying,

$$z = \frac{z_1\sigma_2^2 + z_2\sigma_1^2}{\sigma_1^2 + \sigma_2^2}. \tag{3.9}$$

This estimation above can be also rewritten as,

$$z = z_1 + K\,(z_2 - z_1). \tag{3.10}$$

Where $K = \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}$. And the covariance of the combined result is,

$$\sigma^2 = \frac{\sigma_1^2 \sigma_2^2}{\sigma_1^2 + \sigma_2^2}. \tag{3.11}$$

If we introduce a gain factor *K*, we can rewrite the best estimation of covariance as,

$$\sigma^2 = (1 - K)\sigma_1^2. \tag{3.12}$$

Note that z_1 could not be a measurement. It can be a current belief in the landmark position with a covariance σ_1^2. And z_2 can be a measurement with the error covariance σ_2^2. The Kalman filter would combine the robot current belief with the measurement in order to provide the best possible estimation. Fig 3.1 shows an illustration of how Kalman filter combined two different distributions into one final distribution. Note that the estimation of *z* fulfills the boundary conditions mentioned above. And the covariance σ^2 is smaller than any of σ_1^2 and σ_2^2, which means the combined distribution has a higher belief of where the landmark is. This meets our common sense.

It is easy to expand this 1-D problem to 2-D problem. For static landmarks, their x value and y value in an x-y coordinate are independent. The position on x-axis has nothing relative to the position on y-axis for the same landmark. Therefore, a 2-D Kalman filter in could be divided into two 1-D Kalman filter problem. This method will be used in Chapter 5, where when two robots detected a same landmark, the landmark's posterior could be updated by using the statistic Kalman filter to accurate the position of landmarks as well as to reduce uncertainty of this landmark's covariance.

From the EKF experiment results later in Chapter 4, it could be noticed that when we read the covariance matrix of EKF from the result, there always some "*0*" in the matrix. If we temporarily set the covariance matrix as P here, where

$$P = \begin{bmatrix} P_{robot} & \cdots & \cdots \\ \cdots & P_{landmark1} & \cdots \\ \cdots & \cdots & P_{landmark2} \end{bmatrix}. \quad (3.13)$$

P_{robot} presents a robot covariance which is a 3 x 3 matrix. We do not concern about the robot here. $P_{landmark1}$ and $P_{landmark2}$ are two 2 x 2 matrixes which present landmark 1 and landmark 2's covariance. They can be pressed as

$$P_{landmark1} = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}. \text{and } P_{landmark2} = \begin{bmatrix} c & 0 \\ 0 & d \end{bmatrix}. \quad (3.14)$$

We noticed that only the grid one the diagonal grids has a value (here for *a, b, c,* and *d*), and on the other grids, the value are always 0. This in turn proved that the position x and position y has nothing relative with each other for the same landmark. And we could divide the 2-D Kalman filter problem into two 1-D Kalman filter problem.

3.4 Extend Kalman Filter

The basic Kalman Filter algorithm is the optimal estimator for a linear system with Gaussian noise. The EKF is an extension of the basic Kalman filter to non-linear case. The EKF expresses the non-linear models of motion and measurement models by using the linearized functions that are around the most-likely state of the system. The different between Kalman filter and EKF is that, the KF is used in a statistic situation. For example, when two maps are produced by two robots, then a KF can be used to merge them together. However, when a robot is moving through an environment, the robot's map should be updated by using an EKF.

The motion model will be written as a non-linear function $h(x_{t-1}, \mu_t)$ with linearized noise covariance P_t. And the measurement model will be written as the non-linear function $g(x_t, n_t)$ with linearized noise covariance R_t. The update equation of EKF can be written as follows,

$$\mu_t^- = h(x_{t-1}, \mu_t). \quad (3.15)$$

$$\Sigma_t^- = \Sigma_{t-1} + P_t, \tag{3.16}$$

$$G_x = \nabla_{x_t} g(x_t, n_t)|_{x_t=\mu_t^-; n_t=n_t}, \tag{3.17}$$

$$Z_t = G_x \Sigma_t^- G_x^T + R_t, \tag{3.18}$$

$$\overline{z_{n_t}} = g(\mu_t^-, n_t), \tag{3.19}$$

$$K_t = \Sigma_t^- G_x^T Z_t^{-1}, \tag{3.20}$$

$$\mu_t = \mu_t^- + K_t\left(z_t - \overline{z_{n_t}}\right), \tag{3.21}$$

$$\Sigma_t = (I - K_t G_t)\Sigma_t^-. \tag{3.22}$$

For a complete mathematical introduction the Kalman Filter, see Kalman (1960). Kalman Filter is an optimal mathematic for estimation of linear system. But in real world, the environment is rarely linear. Therefore, it still needs the EKF to estimate the result.

There are two well known problems for EKF-based SLAM. The first problem is quadratic complexity computation. The number of mathematical operations required to consider a control and an observation into the filter. For example, if *k* is the measurement number and n is the state number, then the computational complexity is $O(k^{2.376} + n^2)$ (Montemerlo et al. 2002). For this reason, EKF only works well within small area with fewer landmarks (less than a few hundred). But most real world environments have many more than a few hundred features in them. In these environments, we could see the flaw of EKF.

The second problem with EKF is it has single data association hypothesis per observation. The data association method is based on maximum likelihood heuristic. The problem is that once an incorrect data association is made, it will be never removed. This will possibly make a big mistake. This problem is a minor question in

this thesis. Although it is very important to real robot SLAM algorithm, this thesis focus produces an accuracy estimation of both robot poses and landmarks' positions.

3.5 Particle Filter

The Kalman Filter and the EKF present the probability distribution function (pdf) by using a parameterized model (a multivariate Gaussian). The basic idea of Particle Filter presents the pdf by using a set of sample states, or particles. The density of the samples (particles) in one area of the state space will represent the probability of that region. For example, the area with high probability will contain more particles; while on the other hand, region with low probability will hold fewer or even no particles. If given enough particles, the true pdf can be approximated exactly by using Particle Filter or Monte Carlo approximation.

3.5.1 Implementation in Robot Localization

Here, it gives a short example for robot localization problem by using Particle Filter to illustrate the basic idea that how a Particle Filter works on estimating posterior. The idea of Particle Filter Localization algorithms is to represent the belief by a set of m weighted samples distributed according to the posterior $p(\, s_t, \Theta |\, z^t,\, \mu^t,\, n^t)$, where,

$$p(\, s_t, \Theta |\, z^t,\, \mu^t,\, n^t) \;= \{s^{(i)}, \omega^{(i)}\}_{i=1,2,\ldots m}. \tag{3.23}$$

Each $s^{(i)}$ here is a particle (a state) to present robots potential locations, and $\omega^{(i)}$ is non-negative number, called importance weight, which sum up to one. In the initial pose, particles' importance weights are set uniformly as 1/*m*, where m is the number of particles.

The Particle Filter localization algorithm can be written in three steps:

1. Prediction (Sampling). Sample a state s_t^- by drawing randomly particles s_t^i according to the motion mode $p(\, s_t |\, s_{t-1},\, \mu_t)$. Noise is usually assumed to be

Gaussian, with normal distribution. See Figure 3.2

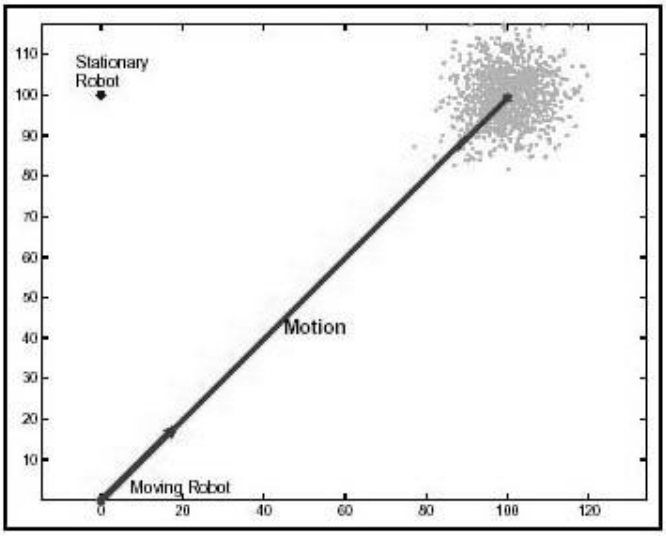


Figure 3.2: The result of prediction (Fox et al. 2000). The robot starts from left-bottom and move to the right-top. Particles on the right-top present the posterior distribution of motion model.

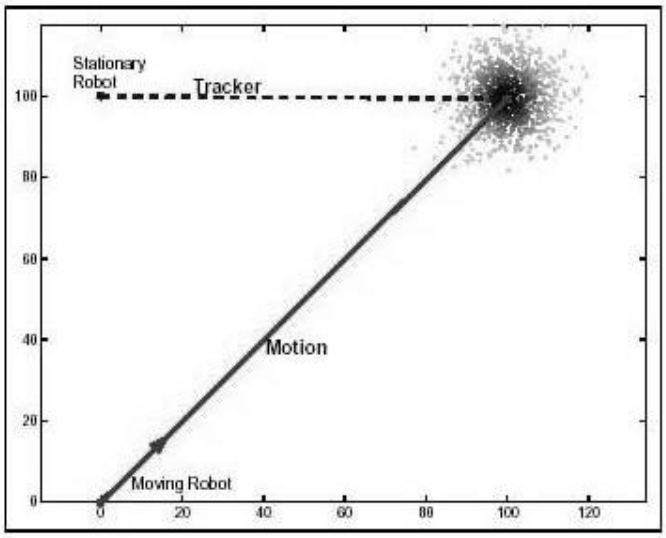


Figure 3.3: The result of update step (Fox et al. 2000). The robot observed a landmark at the top-left, and each particle's weight has been re-assigned according to that measurement. Darker particles present they have higher weights.

2. Update. Each particle's weight is the likelihood of the measurement model $p(z_t|s_t, \Theta, n_t)$ from that particle's hypothesis. See Figure 3.3.

Figure 3.4: The result of resample step (Fox et al. 2000). A new set of particles is chosen so that each particle survives in proportion to its weight. The cloud of resampled particles is more condensed and smoother than un-resampled cloud.

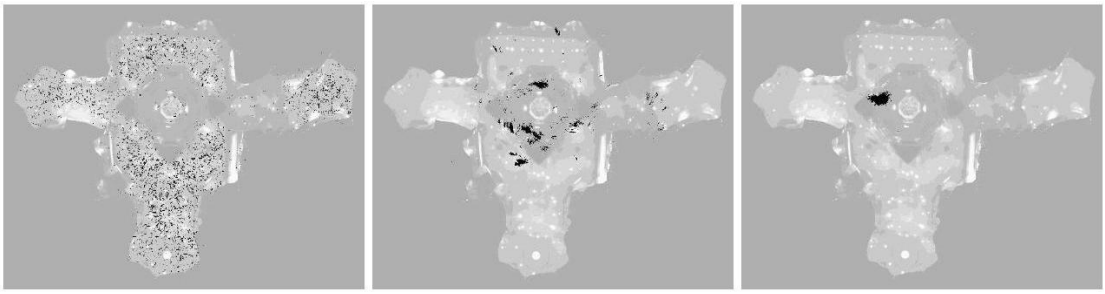
3. Resample. Deleting particles with low weights, and break particles with high weights into more than one particle. The resample step is necessary because most of the particles have drifted enough for their weight to become too small to contribute to the pdf of the moving robot. See Figure 3.4.

   Liu (2008) proposed an effective sample size (ESS) to estimate the number of near-zero-weight particles. When the number of ESS drop below a certain threshold, usually a percentage of the number of particles M, then the particle population is resampled. In this thesis, we set it to *50*%. The ESS is calculated by the following,

$$cv_t^2 = \frac{1}{M}\sum_{i=1}^{M}(M\omega(i) - 1)^2, \tag{3.24}$$

$$ESS = \frac{M}{1+cv^2}. \tag{3.25}$$

   Where M is the number of particles, $\omega(i)$ is the weight for each particle i. *ESS* is the population of effective particles.



a b c

Figure 3.5: The Particle Filter is applied for real robot localization. (a) Initially, particles are randomly set all over the whole map with equal weights. (b) After several sampling and Re-sampling loops, the robot has more confidence on where it is. (c) Finally, it finds its approximate position in the environment (Fox et al. 2000).

Particle filter has been successfully applied in a variety of real world estimation problem in many different fields. One of the most important applications of particle filter in robotics SLAM is Monte Carlo Localization (MCL) (Thrun et al. 2001). In MCL, a set of particles are used to present the possible pose of a robot in a fixed map, shown in Fig 3.5. Not every particle in MCL are equally present the possibility that a robot may stand on that position. Each particle holds a weight to present how important it is, and this means the possibility that a robot may appear on that position.

Unfortunately, the MCL is only used for robot localization. This is because it can not update landmarks' information. Robots must be given enough environment prior knowledge before applying MCL in a real robot. In this case, a much advanced algorithm, called FastSLAM, was developed by Montemerlo (2003). In the following, we will have a look at the FastSLAM algorithm, which is more advance in landmark updating.

3.5.2 FastSLAM

Most of SLAM approaches are based on estimating the posterior over maps and robot pose. It is written as,

$$p(s_t, \Theta | z^t, \mu^t, n^t).$$

But the FastSLAM solution is a little different. It computes the posterior over maps and robot path. As there are many particles to present possible pose of a robot, so we use s^t instead of s_t,

$$p(s^t, \Theta | \, z^t, \, \mu^t, \, n^t).$$

In FastSLAM, the conditional independence is an important consequence. Given knowledge of the robot's path, an observation of one landmark will not provide any information about the position of any other landmark,

$$p(s^t, \Theta | \, z^t, \, \mu^t, \, n^t) = \, p(s^t | \, z^t, \, \mu^t, \, n^t) \prod_{n=1}^{N} p(\Theta_n | s^t, \, z^t, \, \mu^t, \, n^t) \quad (3.26)$$

This factorization, first developed by Doucet et al. (2000), states that the SLAM posterior can be separated into a product of a robot path posterior, and *N* landmark posterior conditioned on the robot's path. The proof of the FastSLAM can be referenced by Montemerlo et al. (2002).

3.5.3 FastSLAM Algorithm

FastSLAM estimates the path posterior $p(s^t, \Theta | \, z^t, \, \mu^t, \, n^t)$ using a modified particle filter. The landmark positions are estimated using EKFs. Therefore, FastSLAM exploits the factored representation by maintaining *MN*+1 filters. All these *MN*+1 filters are low-dimensional and each update of the filter demands a constant computation cost. In total, there are *MN* EKFs, *N* is the number of landmarks, and *M* is the number of particles. Particles in FastSLAM will be denoted as,

$$S_t^i = < \omega_t^i, s_t^i, \mu_{1,t}^i, \Sigma_{1,t}^i, \ldots . \omega_t^i, s_t^i, \mu_{N,t}^i, \Sigma_{N,t}^i >. \quad (3.27)$$

Where *i* is the index of the i^{th} particle, s_t^i is the path of the robot at time *t*, $\mu_{1,t}^i$, and $\Sigma_{1,t}^i$ are the mean and covariance of the Gaussian representing the n^{th} landmark location. The update of FastSLAM is very similar to PF, but there is a little bit different, and the performance is the following steps:

1. Sampling new poses. The same with PF, FastSLAM estimates the new position by drawing a set of sample according to the motion model, and formulate a set of temporary particles

$$s_t^i = p\left(x_t \middle| s_{t-1}^i, u_t\right). \tag{3.28}$$

Here s_t^i is the posterior estimate for the robot pose at time *t*-1 for the i^{th} particle.

2. Updating the observed landmark estimate. Then FastSLAM updates the posterior over each of the landmarks. Landmarks in FastSLAM are represented by a mean value and its covariance. For each of the landmark, if it is not observed, it keeps the mean value and covariance unchanged. If the landmark is observed, it updates according to,

$$p(\Theta_n |\, s^t,\, z^t,\, c^t) = \eta\, p(\, z_t | s^t, \Theta_n, n^t) p(\Theta |\, s^{t-1},\, z^{t-1},\, c^{t-1}). \tag{3.29}$$

If the observation does not correspond to any of the landmarks that have been observed, then a new landmark is added to the map.

For those observed old landmarks, it updates the posterior by using the standard EKF measurement update,

$$\overline{z_t} = g\left(s_t^{[m]}, \mu_{n_t,t-1}\right), \tag{3.30}$$

$$G_{\theta_{n_t}} = \nabla_{\theta_{n_t}} g\left(s_t, \theta_{n_t}\right)\big|_{s_t = s_t^{[m]}; \theta_{n_t} = \mu_{n_t,t-1}^{[m]}}, \tag{3.31}$$

$$Z_{n,t} = G_\theta \Sigma_{n_t,t-1}^{[m]} G_{\theta_{n_t}}^T + R_t, \tag{3.32}$$

$$K_t = \Sigma_{n_t,t-1}^{[m]} G_{\theta_{n_t}}^T Z_{n,t}^{-1}, \tag{3.33}$$

$$\mu_{n_t,t}^{[m]} = \mu_{n_t,t-1}^{[m]} + K_t(z_t - \overline{z_t}), \tag{3.34}$$

$$\Sigma_{n_t,t}^{[m]} = \left(I - K_t G_{\theta_{n_t}}\right) \Sigma_{n_t,t-1}^{[m]}. \tag{3.35}$$

3. Resampling. At last, FastSLAM resample this set of particles. It draws from its temporary particles with replacement according to an importance weight. The necessity to resample arises from the fact that the particle in the temporary set is not distributed according to the desired posterior. It draws only according to the most recent control, assigning no information for the measurement.

   The landmark estimator is an EKF, so this observation likelihood can be computed in closed form. The probability of the observation z_t equal to the probability of the innovation $\mathrm{z_t} - \overline{z_t}^i$ being generated by a Gaussian with zero mean and covariance *Q*. therefore, the weight function can be written as,

$$\omega_t^i = \eta\, |2\pi Q|^{-\frac{1}{2}} exp\,\{-\frac{1}{2}(z_t - \overline{z_t}^i)^T Q_{n,t}^{-1}(z_t - \overline{z_t}^i)\} \tag{3.36}$$

   These three steps together consists the update rule of the FastSLAM algorithm for SLAM problem in a closed form. It is notable that the new samples are only depending on the most recent pose. Consequently, past poses can safely be discarded. Therefore, the FastSLAM does not depend on time *t*, which means it can not remember the long-time trajectory.

3.6 Conclusion

This chapter illustrated the basic methods for EKF-SLAM and FastSLAM to estimate robot pose and landmark positions. The EKF is a mathematical tool to filter out noise from true values. It presents the result by a mean value and a covariance. The mean value means the expected value and the covariance means the uncertainty of this expected value. By using the EKF, we can get the result value close to the true value. FastSLAM can be considered as a combination of *M* independent EKFs, where *M* is the number of particles. For each of the particle, it updates its local map by using the

EKFs (update robot pose and detected landmark positions). But the posterior is presented by a set of particles. The density of particles presents the possible location of robots. The map of the environment is presented by the highest weighted particle's local map.

Both the EKF-based SLAM and Particle-based SLAM (FastSLAM) have their strengths and weakness. The Particle Filter has its strength on non-linear system estimation, and lower computation complexity. Given enough particles, the Particle Filter can exactly estimate the state of system. But it suffers from its tendency sometimes to become overconfident which means that it is unable to succeed estimating the SLAM posterior in a single run. Therefore, a combined Hybrid SLAM method could be formulated by using the strengths and avoiding the weakness of both methods.

4 Hybrid SLAM

Chapter 3 introduced the principles for EKF-based SLAM and Particle Filter based SLAM (FastSLAM). It is known that both methods have their strengths and weakness. In this chapter, it will illustrate a hybrid method of EKF-based SLAM and FastSLAM to estimate the posterior of SLAM, which uses the FastSLAM as front-end to produce maps of environment and uses the EKF as back-end to record robot path. In robot SLAM, the front-end is the process of observe features of map given robot locations. The back-end is the process of localizing robot's pose into a map given a map of environment.

4.1 Introductions to Hybrid Method

The hybrid method SLAM focuses on the problem of feature-based SLAM, where the environment is presented as a set of isolated landmarks. The standard solution is by using a Bayesian approach to calculate the joint probability distributions of robot path and maps. There are two current popular algorithms to modeling this distribution. The first one is EKF-based SLAM, which presents robot pose as a single Gaussian with high-dimension. The other approach is to use Particle Filter to present the robot path by using a set of particles. Examples of this approach include FastSLAM.

Both of these approaches have their strengths and disadvantages. In particular, the EKF-based SLAM is weak for its computational complexity. The computational complexities rise quadrally as the number of landmarks increase. The estimation of robot motion model is linearized, which is not suitable for robot work in real environment. Last, it has difficulty in data association. It requires making hard data association decision to choose the most likely hypothesis. Once a bad data association is made, it cannot revise it.

In contrast to the EKF, the FastSLAM does not suffer from computational complexity

problem and is much more robust in data association. For data association, each particle is allowed to make isolated decisions. As time passes, the particle which had made a bad data association will be removed in the resampling step of Particle Filter. Therefore, the error in data association will be corrected.

The disadvantage of FastSLAM is over confident estimation. This means that in the process of FastSLAM, the region of small area contains more particles. This over confident problem makes FastSLAM difficult to estimate a good map. Another problem is the result of posterior of FastSLAM very much depends on the number of particles. Different sampling particles will produce different uncertainty. For example, when a smaller number is used, the filter will underestimate the total uncertainty because there are not enough particles to represent those areas with low probability. The majority of particles are located the region with high probability that robot position may locate. For expressing whole uncertainty and avoiding over confident, the EKF is far superior. By using a covariance matrix, uncertainty does not decompose. The uncertainty could be a very small value, but it will never be zero when applying the EKF to update robot pose and landmark positions.

Therefore, it is necessary to produce a hybrid filter, which combines the advantages of both approaches. Here, it uses the FastSLAM as a front-end and the EKF-SLAM as back-end. The FastSLAM front-end is used to build local maps, and it is robust enough to do the data association. The EKF-SLAM back-end is used to calculate the position of robots and landmarks in a Gaussian model, which is used for further map jointing for multi-robot SLAM.

4.2 FastSLAM Posterior as Single Gaussian

Before we talk about this problem, it should first have a look at the FastSLAM. The FastSLAM algorithm factors distribution as follows,

$$p(s^t, \Theta | \, z^t, \, \mu^t, \, n^t) = \, p(s^t | \, z^t, \, \mu^t, \, n^t) \prod_{n=1}^{N} p(\Theta_n | s^t, \, z^t, \, \mu^t, \, n^t) \qquad (4.1)$$

Where on the right hand, the first factor presents robot path at time *t*, and the second factor present all landmarks' position given robot pose. The posterior distribution is represented as a set of *M* particles, with each particle, consisting of a robot pose s_t^i, a weight ω, and *N* EKF estimations of landmarks described by the mean $\mu_{n,t}^i$ and covariance $\Sigma_{n,t}^i$,

$$S_t^i = < \omega_t^i, s_t^i, \mu_{1,t}^i, \Sigma_{1,t}^i, \dots \mu_{N,t}^i, \Sigma_{N,t}^i >. \tag{4.2}$$

Alternatively, each particle can equivalently be represented as,

$$S_t^i = < \omega_t^i, x_t^i, P_t^i >. \tag{4.3}$$

Where $x_i^t = [s_t^i, \mu_{1,t}^i, \dots \dots \mu_{N-1,t}^i, \mu_{N,t}^i]^T$ denotes the *i*th particle's robot pose and the mean values of all landmarks, and P_t^i denotes a block-diagonal covariance matrix, which contains robot covariance and the covariance of each landmark, p_t^i is zero in this case, because the particle has no uncertainty about its location,

$$P_t^i = \begin{bmatrix} p_t^i & 0 & 0 & 0 \\ 0 & \Sigma_{1,t}^i & 0 & 0 \\ 0 & 0 & \dots \cdot & 0 \\ 0 & 0 & 0 & \Sigma_{N,t}^i \end{bmatrix} \tag{4.4}$$

By using this representation, the posterior of FastSLAM can be calculated as a single Gaussian with mean value x_t and a covariance P_t according to moment matching,

$$x_t = \sum_{i=1}^{N} \omega_t^i \, x_t^i. \tag{4.5}$$

$$P_t = \sum_{i=1}^{N} \omega_t^i [p_t^i + (x_t^i - x_t)(x_t^i - x_t)^T]. \tag{4.6}$$

where p_t^i is the covariance matrix of robot pose. It sets as 0 at the initial, which means that there is no uncertainty about robot pose at the beginning.

Here, this thesis will take a simple example to illustrate how a FastSLAM posterior could be calculate as a single Gaussian posterior. Assuming at some time, for example at time = *t*, a robot get the posterior by using FastSLAM with two detected landmarks. Therefore, $N = 2$ in the equation (4.2). And then we assume the population of particles is 3 (3 particles is not enough to acquire an accurate map in real world or even in a simulation level on computer, but here we just set this number for examples). Then the equation 4.2 could be expressed as,

$$S_t^1 = < \omega_t^1, \begin{bmatrix} x_t^1 \\ y_t^1 \\ \theta_t^1 \end{bmatrix}, \begin{bmatrix} x_{1,t}^1 \\ y_{1,t}^1 \end{bmatrix}, \begin{bmatrix} \Sigma_{1,x,\mathrm{t}}^1 & 0 \\ 0 & \Sigma_{1,y,\mathrm{t}}^1 \end{bmatrix}, \begin{bmatrix} x_{2,t}^1 \\ y_{2,t}^1 \end{bmatrix}, \begin{bmatrix} \Sigma_{2,x,\mathrm{t}}^1 & 0 \\ 0 & \Sigma_{2,y,\mathrm{t}}^1 \end{bmatrix} >. \quad (4.7)$$

In equation 4.7, $[x_t^1, y_t^1, \theta_t^1]^{\mathrm{T}}$ presents first particle's location at time *t*. $[x_{1,t}^1, y_{1,t}^1]^{\mathrm{T}}$ and $[x_{2,t}^1, y_{2,t}^1]^{\mathrm{T}}$ present the first and the second landmarks position in first particle. $\begin{bmatrix} \Sigma_{1,x,\mathrm{t}}^1 & 0 \\ 0 & \Sigma_{1,y,\mathrm{t}}^1 \end{bmatrix}$ and $\begin{bmatrix} \Sigma_{2,x,\mathrm{t}}^1 & 0 \\ 0 & \Sigma_{2,y,\mathrm{t}}^1 \end{bmatrix}$ are the covariance for each particle. For particle number 2 and particle number 3 it could be documented as S_t^2 and S_t^3. Where:

$$S_t^2 = < \omega_t^2, \begin{bmatrix} x_t^2 \\ y_t^2 \\ \theta_t^2 \end{bmatrix}, \begin{bmatrix} x_{1,t}^2 \\ y_{1,t}^2 \end{bmatrix}, \begin{bmatrix} \Sigma_{1,x,\mathrm{t}}^2 & 0 \\ 0 & \Sigma_{1,y,\mathrm{t}}^2 \end{bmatrix}, \begin{bmatrix} x_{2,t}^2 \\ y_{2,t}^2 \end{bmatrix}, \begin{bmatrix} \Sigma_{2,x,\mathrm{t}}^2 & 0 \\ 0 & \Sigma_{2,y,\mathrm{t}}^2 \end{bmatrix} >. \quad (4.8)$$

$$S_t^3 = < \omega_t^3, \begin{bmatrix} x_t^3 \\ y_t^3 \\ \theta_t^3 \end{bmatrix}, \begin{bmatrix} x_{1,t}^3 \\ y_{1,t}^3 \end{bmatrix}, \begin{bmatrix} \Sigma_{1,x,\mathrm{t}}^3 & 0 \\ 0 & \Sigma_{1,y,\mathrm{t}}^3 \end{bmatrix}, \begin{bmatrix} x_{2,t}^3 \\ y_{2,t}^3 \end{bmatrix}, \begin{bmatrix} \Sigma_{2,x,\mathrm{t}}^3 & 0 \\ 0 & \Sigma_{2,y,\mathrm{t}}^3 \end{bmatrix} >. \quad (4.9)$$

Alternatively, S_t^i could equivalently be represented as,

$$S_t^1 = < \omega_t^1, \begin{bmatrix} x_t^1 \\ y_t^1 \\ \theta_t^1 \\ x_{1,t}^1 \\ y_{1,t}^1 \\ x_{2,t}^1 \\ y_{2,t}^1 \end{bmatrix}, \begin{bmatrix} 0 & & & & & & \\ & 0 & & & & & \\ & & 0 & & & & \\ & & & \Sigma_{1,x,\mathrm{t}}^1 & & & \\ & & & & \Sigma_{1,y,\mathrm{t}}^1 & & \\ & & & & & \Sigma_{2,x,\mathrm{t}}^1 & \\ & & & & & & \Sigma_{2,y,\mathrm{t}}^1 \end{bmatrix} >. \quad (4.10)$$

The same as particle number 2 and number 3, where

$$S_t^2 = < \omega_t^2, \begin{bmatrix} x_t^2 \\ y_t^2 \\ \theta_t^2 \\ x_{1,t}^2 \\ y_{1,t}^2 \\ x_{2,t}^2 \\ y_{2,t}^2 \end{bmatrix}, \begin{bmatrix} 0 & & & & & & \\ & 0 & & & & & \\ & & 0 & & & & \\ & & & \Sigma_{1,x,\mathrm{t}}^2 & & & \\ & & & & \Sigma_{1,y,\mathrm{t}}^2 & & \\ & & & & & \Sigma_{2,x,\mathrm{t}}^2 & \\ & & & & & & \Sigma_{2,y,\mathrm{t}}^2 \end{bmatrix} >. \quad (4.11)$$

$$S_t^3 = < \omega_t^3, \begin{bmatrix} x_t^3 \\ y_t^3 \\ \theta_t^3 \\ x_{1,t}^3 \\ y_{1,t}^3 \\ x_{2,t}^3 \\ y_{2,t}^3 \end{bmatrix}, \begin{bmatrix} 0 & & & & & & \\ & 0 & & & & & \\ & & 0 & & & & \\ & & & \Sigma_{1,x,\mathrm{t}}^3 & & & \\ & & & & \Sigma_{1,y,\mathrm{t}}^3 & & \\ & & & & & \Sigma_{2,x,\mathrm{t}}^3 & \\ & & & & & & \Sigma_{2,y,\mathrm{t}}^3 \end{bmatrix} >. \quad (4.12)$$

On the next, we could calculate the weight mean of these three particle's location to present robot's pose and landmarks' position at time *t*.

$$x_t = \omega_t^1 \cdot \begin{bmatrix} x_t^1 \\ y_t^1 \\ \theta_t^1 \\ x_{1,t}^1 \\ y_{1,t}^1 \\ x_{2,t}^1 \\ y_{2,t}^1 \end{bmatrix} + \omega_t^2 \cdot \begin{bmatrix} x_t^1 \\ y_t^1 \\ \theta_t^1 \\ x_{1,t}^1 \\ y_{1,t}^1 \\ x_{2,t}^1 \\ y_{2,t}^1 \end{bmatrix} + \omega_t^3 \cdot \begin{bmatrix} x_t^3 \\ y_t^3 \\ \theta_t^3 \\ x_{1,t}^3 \\ y_{1,t}^3 \\ x_{2,t}^3 \\ y_{2,t}^3 \end{bmatrix}. \quad (4.13)$$

Then the uncertainty could be calculated,

$$P_t = \omega_t^1 \cdot \left\{ \begin{bmatrix} 0 & & & & & & \\ & 0 & & & & & \\ & & 0 & & & & \\ & & & \Sigma_{1,x,\mathrm{t}}^1 & & & \\ & & & & \Sigma_{1,y,\mathrm{t}}^1 & & \\ & & & & & \Sigma_{2,x,\mathrm{t}}^1 & \\ & & & & & & \Sigma_{2,y,\mathrm{t}}^1 \end{bmatrix} + \begin{bmatrix} \begin{bmatrix} x_t^1 \\ y_t^1 \\ \theta_t^1 \\ x_{1,t}^1 \\ y_{1,t}^1 \\ x_{2,t}^1 \\ y_{2,t}^1 \end{bmatrix} - x_t \end{bmatrix} \right.$$
$$\left. \cdot \begin{bmatrix} \begin{bmatrix} x_t^1 \\ y_t^1 \\ \theta_t^1 \\ x_{1,t}^1 \\ y_{1,t}^1 \\ x_{2,t}^1 \\ y_{2,t}^1 \end{bmatrix} - x_t \end{bmatrix}^T \right\} + \{\omega_t^2 \cdot [p_t^2 + (x_t^2 - x_t)(x_t^2 - x_t)^T\} + \{\omega_t^3$$
$$\cdot [p_t^3 + (x_t^3 - x_t)(x_t^3 - x_t)^T\}.$$

(4.14)

In this example, the covariance matrix is a 7×7 matrix, with first three left top matrix presents the robot's pose uncertainty.

After these three steps, the FastSLAM posterior with particles and weight has been summed together with a Gaussian posterior.

4.3 Hybrid Method SLAM Algorithm

In real world SLAM applications, data association problem is the most difficult. However, this data association is not the main interest of this thesis. For this reason, this hybrid method algorithm is assumed that correct data association is made.

The algorithm first predicts and updates robot pose and landmarks by using standard FastSLAM, in which, the robot pose posterior is estimated by using a particle filter. The landmark posteriors are estimated by using EKFs. Each particle records a set of landmarks of its own by using EKFs. Different particles estimate and update the posterior of landmarks dependently to each other. This part of algorithm is very

similar to FastSLAM.

When the importance weight is calculated, in FastSLAM algorithms, the representation of robot path and location of landmarks are selected as the position of particles with the highest weight. In the hybrid SLAM algorithm, the position of robot path and landmarks are presented by the weight mean of each particle, and covariance is the sum of each particle's difference between the weight mean value and each particle's position value, multiplying its weight.

The last step of hybrid SLAM algorithm is re-initialization. All particles' information should be reset as the calculated new values. In each particle, the estimated position of robot and landmarks are reset as the weighted mean of all particles, as well as the covariance of each particle. Note that, in FastSLAM, there is no covariance concerning to robot pose. The uncertainty of robot pose is represented by different particles and their weights. However in hybrid SLAM, the uncertainties of robot pose influences the robot's observations for the next time. Therefore, we add this uncertainty to the observation noise for the next time step.

4.3.1 Sampling a New Pose

The first step of the hybrid SLAM is to estimate the robot's new pose at time *t*, given the information s_{t-1} from last time. This information contains robot pose, particle's weight, all detected landmark positions, and their uncertainties (covariance matrix) .This estimation of a new robot pose for time *t* is obtained by sampling from the probabilistic motion model by using particle filter. As this part of algorithm is the same with FastSLAM, the motion model noise could be set with any non-Gaussian function. But in this thesis, we still use Gaussian noise for motion noise.

x_{t-1} is robot pose for time t-1, u_t is the control input, with velocity and turnings, Q_t is the control noise at time t, with $Q_t(1,1)$ presents the transform error and $Q_t(2,2)$ presents the turning error. M is the population of particles. x_{t-1} is robot pose at time *t*-1, which is $[x, y, \theta]^T$. The output of this algorithm is the new particles' position,

which presents a potential location of a robot at time *t*.

Algorithm for sampling a new pose (x_{t-1}, $\mathbf{u_t}$, $\mathbf{Q_t}$)
1: for i = 1 to M % M particles in total
2: $x_t^i = x_{t-1} + \begin{bmatrix} [u_t(1,1) + \text{sqrt}(Q(1,1))] \cdot \cos(u_t(2,1) + \text{sqrt}(Q_t(2,2))) \\ [u_t(1,1) + \text{sqrt}(Q_t(1,1))] \cdot \sin(u_t(2,1) + \text{sqrt}(Q_t(2,2))) \\ u_t(2,1) + \text{sqrt}(Q_t(2,2)) \end{bmatrix}$
3: endfor
4: return $x_t^{i:M}$

4.3.2 Updating the Landmark Estimates

In the SLAM approach, the landmarks are conditioned on the robot's path. Therefore, landmark positions can not be easily obtained by robot's observation information, as sensor readings are always affected by noise. One of the most important things for robot SLAM is to update and revise the estimated landmark positions. As the Hybrid method uses FastSLAM as front-end, the landmark updating step is exactly the same with FastSLAM.

For those re-observed landmarks, landmark position is updated by an EKF. The basic idea is that as we know the position of the re-observed landmark at time t-1, and we know the new particle's position by using **sampling a new pose ($\mathbf{x_{t-1}}$, $\mathbf{u_t}$, $\mathbf{R_t}$) algorithm**, we could compute the estimated distance and bearing between the particle and the observed landmark. This is $\overline{z_t}$.

Then the Jacobians G_n of observation model could be calculated by

$$G_n = \begin{bmatrix} \frac{\theta_{n,x} - s_{t,x}}{\sqrt{q}} & \frac{\theta_{n,y} - s_{t,y}}{\sqrt{q}} \\ -\frac{\theta_{n,y} - s_{t,y}}{q} & \frac{\theta_{n,x} - s_{t,x}}{q} \end{bmatrix}. \quad (4.15)$$

$$q = (\theta_{n,x} - s_{t,x})^2 + (\theta_{n,y} - s_{t,y})^2. \quad (4.16)$$

Where the robot pose at time t is presented as $< s_{t,x}, s_{t,y}, s_{t,\theta} >$, and the current landmark position is $<\theta_{n,x}, \theta_{n,y}>$. n is the landmark order. q is the distance between robot and the current observed landmark. The Jacobians generalizes the gradient of a scalar valued function from current robot pose to the observed landmark position. Or it can be considered as the uncertainty change rate along with the range. Then the observation uncertainty can be calculated by using

$$Z_{n,t} = G_\theta \Sigma^{[m]}_{n_t,t-1} G^T_{\theta_{n_t}} + R_t. \quad (4.17)$$

Where R_t is observation noise, which is a Gaussian noise. $\Sigma^{[m]}_{n_t,t-1}$ is the nth landmark's uncertainty at time t-1. Then we could get the updated position of landmark as well as its covariance. First we calculate the Kalman gain, then we use this gain to update landmark's information.

$$K_t = \Sigma^{[m]}_{n_t,t-1} G^T_{\theta_{n_t}} Z^{-1}_{n,t}, \quad (4.18)$$

$$\mu^{[m]}_{n_t,t} = \mu^{[m]}_{n_t,t-1} + K_t(z_t - \overline{z_t}), \quad (4.19)$$

$$\Sigma^{[m]}_{n_t,t} = \left(I - K_t G_{\theta_{n_t}}\right) \Sigma^{[m]}_{n_t,t-1}. \quad (4.20)$$

For those unobserved landmarks, the SLAM algorithm keeps all information the same with their previous time step. The mean values of position and the same values of covariance matrix are kept unchanged.

It is notable that, when a new landmark is observed, the algorithm should give this landmark a covariance to represent its uncertainty. There is a very simple way of doing this. Instead of computing the correct initial covariance, the covariance can be set to a high value (Thrun et at, 1998). Either use these following methods, or just

manually set a number. In this thesis, it set this value at 30 meters to present the covariance. The value 30 is big enough to cover all possible landmark positions when the robot first observes this landmark.

Algorithm for update landmark information ($s_{t-1}, s_t, \mathrm{r_t}, \phi_\mathrm{t}, \theta_n, \Sigma_{n_t,t-1}$)

1: for all observed landmarks, do

2: if landmark never seen before

3: $$\begin{pmatrix}\theta_{n,x}\\ \theta_{n,y}\end{pmatrix} = \begin{pmatrix}s_{t,x}\\ s_{t,y}\end{pmatrix} + \mathrm{r_t}\begin{pmatrix}\cos(\mathrm{s_{t,\theta}} + \phi_\mathrm{t})\\ \sin(\mathrm{s_{t,\theta}} + \phi_\mathrm{t})\end{pmatrix}$$

4: endif

5: $$\mathrm{G_n} = \begin{bmatrix}\frac{\theta_{n,x}-s_{t,x}}{\sqrt{q}} & \frac{\theta_{n,y}-s_{t,y}}{\sqrt{q}}\\ -\frac{\theta_{n,y}-s_{t,y}}{q} & \frac{\theta_{n,x}-s_{t,x}}{q}\end{bmatrix}$$

6: $$\mathrm{q} = (\theta_{n,x} - s_{t,x})^2 + (\theta_{n,y} - s_{t,y})^2$$

7: $$\overline{z_t} = \begin{pmatrix}\sqrt{q}\\ \mathrm{atan2}(\theta_{n,x} - s_{t,x}, \theta_{n,y} - s_{t,y})\end{pmatrix}$$

8: $$\mathrm{K_t} = \Sigma_{n_t,t-1}\mathrm{G_n}\mathrm{Z}_{\mathrm{n,t}}^{-1}$$

9: $$\mathrm{s}_{\mathrm{n_t,t}}^{[m]} = \mathrm{s}_{\mathrm{n_t,t-1}}^{[m]} + \mathrm{K_t}(\mathrm{z_t} - \overline{\mathrm{z_t}})$$

10: $$\Sigma_{n_t,t} = (\mathrm{I} - \mathrm{K_t}\mathrm{G_n})\Sigma_{\mathrm{n_t,t-1}}$$

11: endfor

12: return $(\theta_n, \Sigma_{n_t,t}^{[m]})$

In this algorithm, $\mathrm{s_{t-1}}, \mathrm{s_t}$ are robot pose at time *t*-1 and *t*. $\mathrm{r_t}$, and ϕ_t are the sensing readings, with $\mathrm{r_t}$ presents the distance between a robot and the observed landmark while ϕ_t presents the angle between a robot and the observed landmark. θ_n and $\Sigma_{\mathrm{n_t,t-1}}$ are landmarks location value and its covariance matrix at

time *t*-1. The output of this algorithm is the updated landmark information in terms of feature's location and its uncertainties.

4.3.3 Calculating Importance Weights

The most important thing in FastSLAM is particle's weight, because the SLAM posterior in Particle Filter is represented by a set of particles with different weights. In HybridSLAM, the SLAM posterior is represented by a mean value and a covariance matrix. But these values are calculated by using particles' position and their weights. In this section, it will illustrate how to compute particles' weights. *m*th particle's weight could be calculated by:

$$\omega_t^{[m]} = \frac{1}{\sqrt{|2\pi Z_{n,t}|}} \exp\left\{-\frac{1}{2}\left(z_t - \overline{z_{n,t}}\right)^T \left[Z_{n,t}\right]^{-1}\left(z_t - \overline{z_{n,t}}\right)\right\}. \qquad (4.21)$$

4.3.4 Calculating the Single Gaussian Posterior

For each time of step, all particles' information should be summed together to get robot pose and the features' locations as well as their uncertainty by using equation (4.5) and (4.6). This summed result is then used as the initial information for the next time step. A simple example has been given in previous section, and it is not difficult to do this.

As none of the FastSLAM update equations depend on the total path and only the most recent pose is used to update the particle set, particles could forget all information before time *t*-1. Fortunately, the Hybrid method SLAM can help FastSLAM to remember its data before time *t*-1.

4.4 The Experiment

This thesis setups experiments to compare three different SLAM filters to compare their estimation ability. The experiments focus on algorithms' ability to correct estimate robot's pose and landmark positions in a simulation environment. This was done by asking one robot moving with the same trajectory (the same control input)

and the same environment (the same landmarks positions and the same noise for both control and sensing) but applying different algorithms. Robot poses errors over the whole pass and a landmark position will be compared. In this experiment, data association is assumed to be known.

4.4.1 Filters

Experiments compared the following filters in simulation level:

1. EKF-SLAM algorithm which is described in Chapter 3,

2. FastSLAM algorithms which is described in Chapter 3, and

3. HybridSLAM algorithm which is a combination of FastSLAM and EKF SLAM algorithm.

4.4.2 Environment

The experimental environment consisted of 500 steps of movement in a simulated 2-dimensional world of size 100m x 100m. This simulated world contains 18 unique landmarks. Robots move in a constant speed of approximately 0.47m per step (in this simulation algorithms, robot true is obtained by left click the mouse, robot whole path will first be summed together, and then divided by 500, finally it will calculate robot moving speed). The turning angle is from $0°/\text{step}$ to $100°/\text{step}$. It uses a $360°$ range sensor with a maximum range of 20m. The particle number is 50 in this experiment. Robot control noise in distance is set as 5cm, and the noise in angle is set as 3°. The error for the range find is 2m in distance and 10°.

When robots start in this experiment, is assumed robots known its starting pose with a high certainty.

4.4.3 Results

Figure 4.1 illustrates the SLAM result for EKF algorithm. The expected robot path is indicated by green line (path), the red stars in this map are the true positions for

landmarks. The red line (path) is the estimated path for robot thought where they are in the map. The blue spots in the map around the red stars are the estimated landmark positions. The robot starts at left-bottom which is at about (10, 40) in the map, and stops at about (70, 70) in the map.

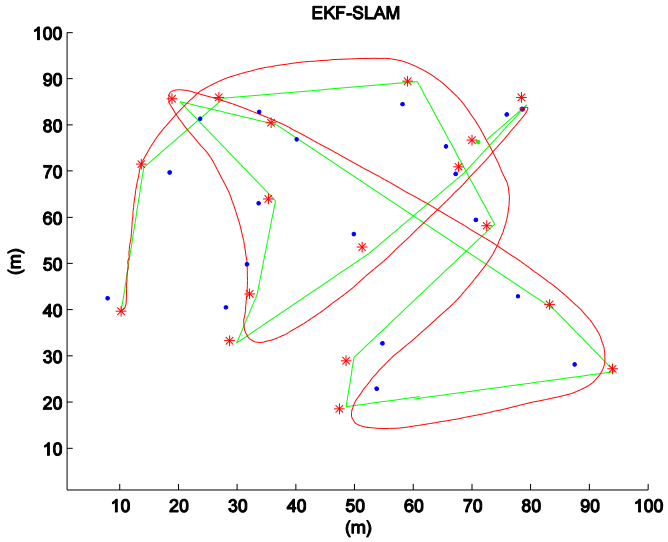


Figure 4.1: EKF-SLAM 2-D simulation result. Green line is the true path, and red line is the estimated path of robot thinking where it is. Red stars are the true positions

Figure 4.2 illustrates the SLAM result for FastSLAM algorithm. The expected robot path is indicated by red line (path), the red stars in this map are the true positions for landmarks. The green line (path) is the estimated path for robot thought where they are in the map. The blue spots in the map around the red stars are the estimated landmark positions. The robot starts at left-bottom which is at about (10, 40) in the map, and stops at about (70, 70) in the map.

From the results, we can see that on the right-bottom side of the map, the estimated robot path is suddenly far from the true path with a large error. This happens probably because of the FastSLAM's overconfidence. For some reasons, particles in FastSLAM are sampled in a certain area where it is far from the true path. And FastSLAM can only remember their path for a short time (from t-1 to t), this mistake can not be corrected immediately. Therefore, this error lasts for some time.

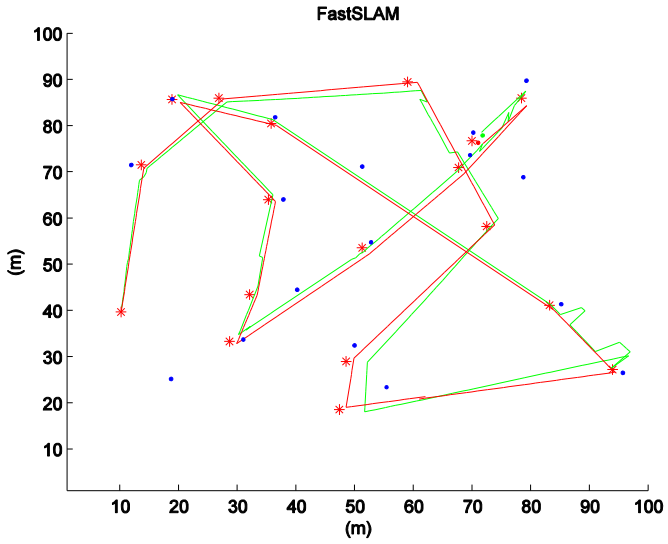


Figure 4.2: FastSLAM 2-D simulation result. Red line is the true path, and green line is the estimated path of robot thinking where it is. Red stars are the true positions and blue spots are the estimated robot landmarks.

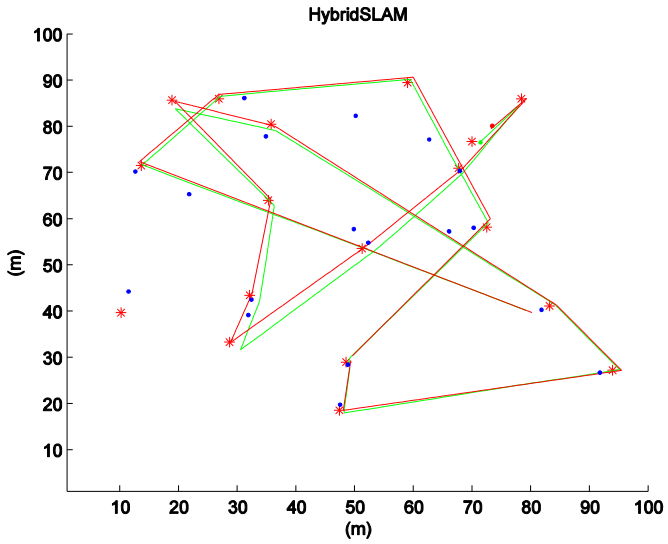


Figure 4.3: HybridSLAM 2-D simulation result. Red line is the true path, and green line is the estimated path of robot thinking where it is. Red stars are the true positions and blue spots are the estimated robot landmarks.

Figure 4.3 illustrates the SLAM result for FastSLAM algorithm. The expected robot path is indicated by red line (path), the red stars in this map are the true positions for landmarks. The green line (path) is the estimated path for robot thought where they are in the map. The blue spots in the map around the red stars are the estimated

landmark positions. The robot starts at right-bottom which is at about (80, 40) in the map, and stops at about (70, 70) in the map.

From this result, we can see that robot path is smoother than FastSLAM. There is no such “jump” in the map, and the average error between true landmark positions and estimated landmark positions is smaller than EKF SLAM.

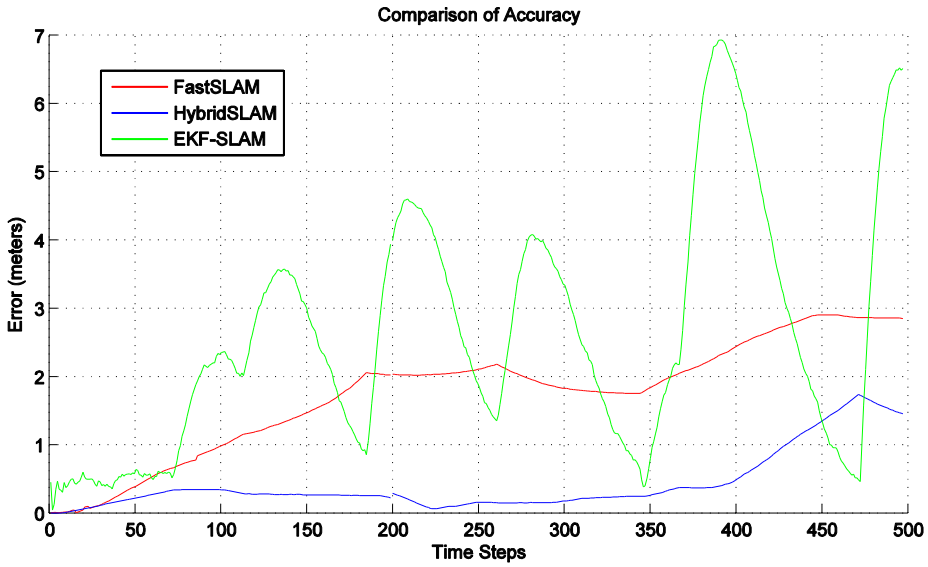


Figure 4.4: A comparison of pose accuracy by three SLAM algorithms

The pose accuracy of Hybrid method SLAM was compared with that of the EKF SLAM and FastSLAM algorithms on computers (shown in Figure 4.4). The error of the EKF is shown in green color. Red line shows the errors of FastSLAM estimation and the blue line represents estimation errors of HybridSLAM. The x-axis is the true path, which means zero error.

In this experiment, filters estimation error increases with time. However, the hybrid SLAM method has a lower error in robot pose estimation, with a maximum error of 2 meters. The EKF SLAM algorithm estimation error could be 7 meters at maximum, and the error performed by FastSLAM algorithm is about 3 meters at maximum. Before 400 time steps, the hybrid SLAM's error is quite small, with the error within 0.5 meter. This experiment suggests that the hybrid SLAM algorithm has a better

performance compared to EKF SLAM algorithm and FastSLAM algorithm, and in a relevant small distance, the HybridSLAM algorithm has a very small estimation error.

The performance of HybridSLAM algorithm on different terrain was compared with that of FastSLAM algorithm in the experiment. In this simulation, different Terrain is presented by different levels of odometric noise (with high level odometric noise = 50cm in distance and low level odometric noise = 5cm in distance; with high level odometric noise 3° in heading and low level odometric noise = 10° in heading). The result of the HybridSLAM algorithm and FastSLAM algorithms given low and high levels of odometric noise are shown in Figure 4.5(a), (b), (c) and (d). The number of landmarks and landmarks' position is the same with that from Figure 4.1 to Figure 4.3. Exact positions could be found in the Appendix. But here, this thesis deletes those landmarks from the map, because this experiment only concerns about robot's path.

Both algorithms correct estimated robot path when the odometric noise is low. But the FastSLAM fails when the odometric noise is high. The FastSLAM algorithm localizes the robot in an incorrect location often. From this experiment, it shows that the HybridSLAM could be used in a complicated terrain rather than the FastSLAM algorithm.

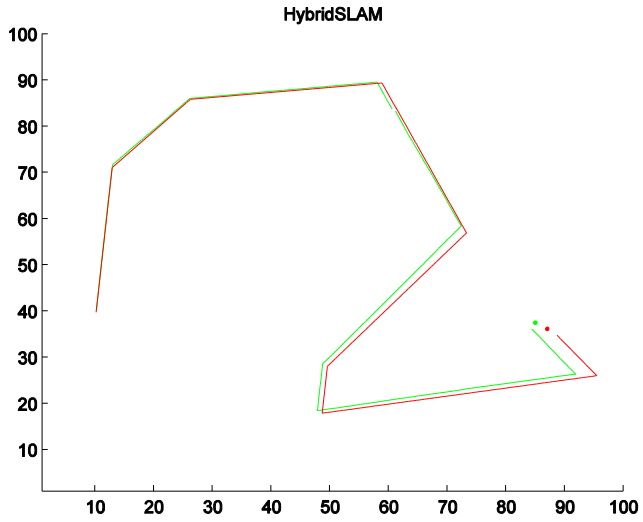


Figure 4.5 (a) HybridSLAM with high odometry error

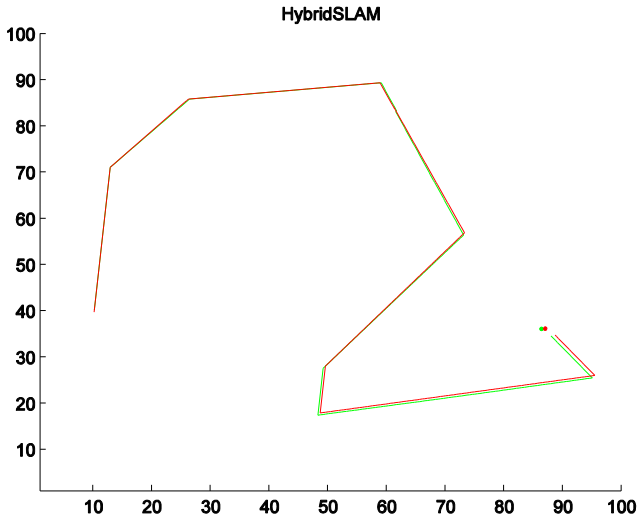


Figure 4.5 (b) HybridSLAM with high odometry errors

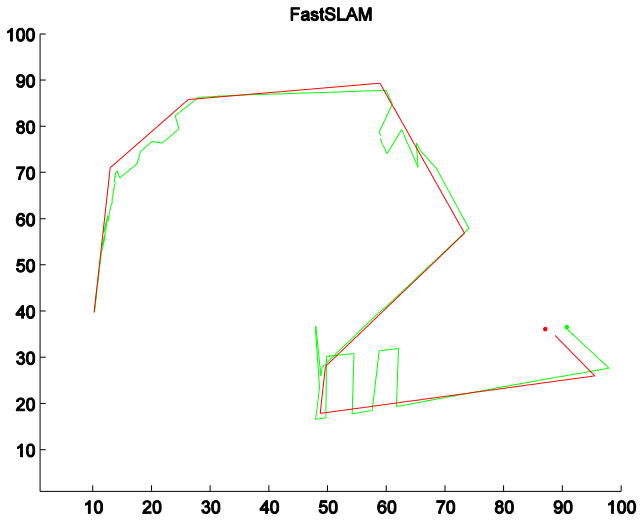


Figure 4.5 (c) FastSLAM with high odometry error

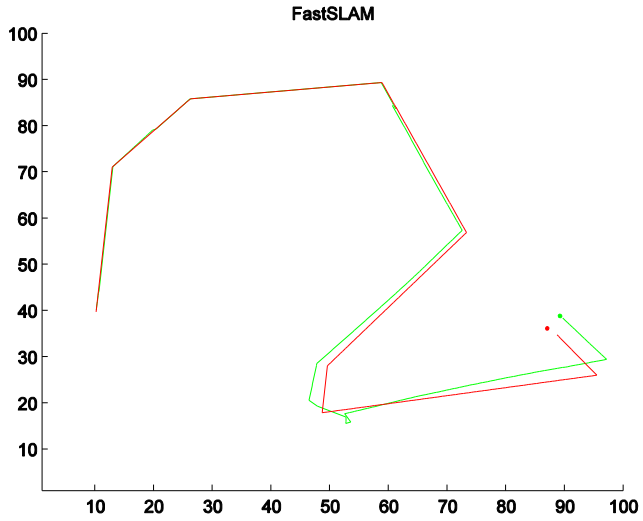


Figure 4.5 (d) FastSLAM with low odometry error

Another experiment is set to evaluate the errors in landmark estimation for different SLAM algorithm. The map of environment is represented by different landmarks in this thesis. Therefore, the performance in landmark estimation presents the accuraty of map. In Figure 4.6, there is a comparison of estimation errors are made by EKF, FastSLAM and HybridSLAM algorithms. The red line indicates the estimation errors of FastSLAM. Green line is the result errors of EKF-SLAM, and blue line is Hybrid-SLAM errors for landmarks.

In this experiment, a robot detected this landmark at its initial time, and then robot left this landmark at the 45th steps (out of the maximum detecting range for sensors). After that time, the estimated position of landmarks keeps the same, as well as landmarks covariance. When the robot first detected this landmark, there was a big error in all EKF-SLAM, FastSLAM, and Hybrid SLAM algorithms as we set the initial uncertainty value are very high. Then as time goes by, all of these three algorithms successfully reduce this error within 5 meters. But the EKF-SLAM algorithm performs better. When robot left this landmark, the estimation error of EKF-SLAM algorithm is the smallest, with about 2 meters.

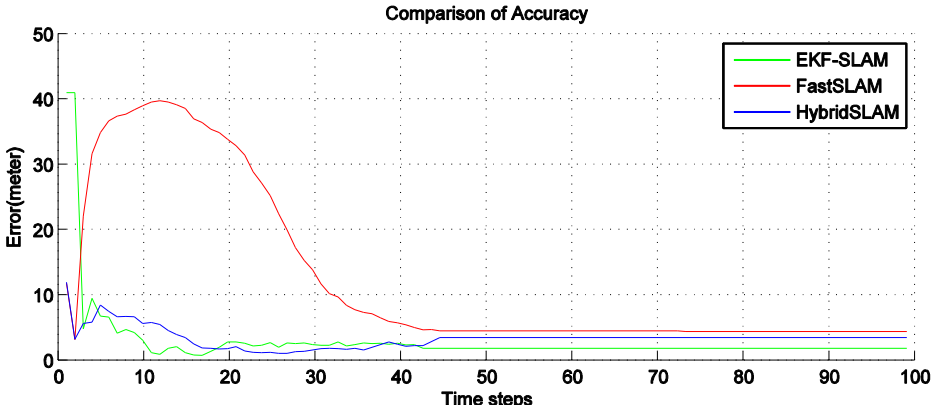


Figure 4.6: The comparison of accuracy in landmark positions. The landmark is chosen as the first landmark that robot detected

4.5 Conclusion

This chapter illustrated the basic Hybrid method for robot SLAM. Generally, the Hybrid SLAM posterior is the weight mean of each particle in FastSLAM. Robot poses and landmark positions are presented by mean values and their covariance. The mean value of robot pose is also used as its starting pose for the next time step. The advantage of this method to present robot path is that it look smoother compared to FastSLAM. It clearly showed in the experiment that robot path for FastSLAM will go far away from its true path and then suddenly jump back around the true path. This is because the particle used to present robot pose and landmarks is changed. This situation does not happen in EKF-SLAM and Hybrid-SLAM. We also found that Hybrid SLAM has a smaller error in estimating robot pose, and this estimation is not affected by odometry noise as much as FastSLAM is.

5 Multi-Hybrid SLAM

Most SLAM problems are done by using a single robot, but there is some research that has been conducted on using team robot for the SLAM problem. Chapter 4 has illustrated the basic Hybrid SLAM algorithm. This thesis proposed a new multi-SLAM algorithm, which is called multi Hybrid SLAM. This chapter will illustrate the basic idea for multi-robot Hybrid SLAM algorithm, and give some experiment results on multi-Hybrid SLAM algorithm compared with multi-FastSLAM algorithm.

5.1 Multi-Hybrid SLAM

When applying the multi-Hybrid SLAM, robots are assumed to conduct the SLAM problem individually. A limited senor is equipped on the robot, which used to sense landmarks or other member robots. When a robot senses another robot, they can start to transfer their mapping information to each other. In this thesis, the mapping information involves landmark positions and the positions' covariance. The distance and bearing could be known between two robots when they are meeting. This thesis here only considers two robots SLAM situation with the data association is known.

The basic idea for this multi-hybrid SLAM is to merge landmarks on different maps. One landmark could be located on different positions in different maps that draw by different robots. This is because of noise that either comes from robot itself or comes from the environment. Kalman filter is known as one of the most powerful mathematic tool to handle noise problem. The algorithm will use Kalman filter to merge the same landmark in different maps. The solution has been discussed in Chapter 3.

For example, when two robots meet at a time, one landmark Θ is presented by Θ^{A}

and Θ^{B} in different robot's map, as well as their covariance $\Sigma^{A} = \begin{bmatrix} \Sigma_x^A & \\ & \Sigma_y^A \end{bmatrix}$ and $\Sigma^{B} = \begin{bmatrix} \Sigma_x^B & \\ & \Sigma_y^B \end{bmatrix}$. Then estimating this landmark position could be considered as two 1-D Kalman filter problem for x-axis and y-axis.

Then we can calculate the new position for this landmark:

$$\Theta_x = \Theta_x^A + \frac{\Sigma_x^A}{\Sigma_x^A + \Sigma_x^B} \cdot (\Theta_x^A - \Theta_x^B). \tag{5.1}$$

$$\Sigma_x = \frac{\Sigma_x^A \cdot \Sigma_x^B}{\Sigma_x^A + \Sigma_x^B}. \tag{5.2}$$

$$\Theta_y = \Theta_y^A + \frac{\Sigma_y^A}{\Sigma_y^A + \Sigma_y^B} \cdot \left(\Theta_y^A - \Theta_y^B\right). \tag{5.3}$$

$$\Sigma_y = \frac{\Sigma_y^A \cdot \Sigma_y^B}{\Sigma_y^A + \Sigma_y^B}. \tag{5.4}$$

By using the Kalman filter, we could combine those landmarks' position from different maps to find a new position for those landmarks.

5.2 Experiment

5.2.1 Filters

Experiments compared the following filters in simulation level:

1. Single Hybrid SLAM which has been discussed in Chapter 4,

2. Multi-FastSLAM algorithms, and

3. Multi-Hybrid SLAM algorithm.

Experiments are set to compare the estimation accuracy on robot path and landmarks positions. In this experiment, a robot or robots randomly move through an open area

perform the three algorithms on computers. Robot path and landmarks positions are record for this comparison.

5.2.2 Environment

The experimental environment consisted of 500 steps of movement in a simulated 2-dimensional world of size 100m x 100m. This simulated world contains 18 unique landmarks. Robot speed and turning angles are not set as constant value because the robot randomly moves in this simulation area. It uses a 360° range sensor with a maximum range of 20m. The particle number is 50 for FastSLAM and Hybrid SLAM. Robot control noise in distance is set as 5cm, and the noise in angle is set as 3°. The error for the range find is 2m in distance and 10°. Robot initial pose is known.

5.2.3 Results

This section will present experimental results to validate the performance of the multi-Hybrid SLAM algorithm. Figure 5.1 is the experiment result for multi-Hybrid SLAM. This is a whole process from time 0 to time = 500 (the end). We record some maps at time = 20, time = 100, time = 200 and time = 500. Red paths are the true trajectories and green paths are the estimated paths. Blue spots are the estimated position of landmarks, and red stars are the true position of landmarks. Robot 1 starts at the position of (40, 80) and moves to the left. Robot 2 starts at the position of (10, 40) and moves to the right.

At the beginning, two robots only detect 6 landmarks. Therefore only those landmarks are detected have an estimated position which is presented by a blue spot. With time goes on, more and more landmarks are detected by both robots. And robot poses are updated by those landmarks.

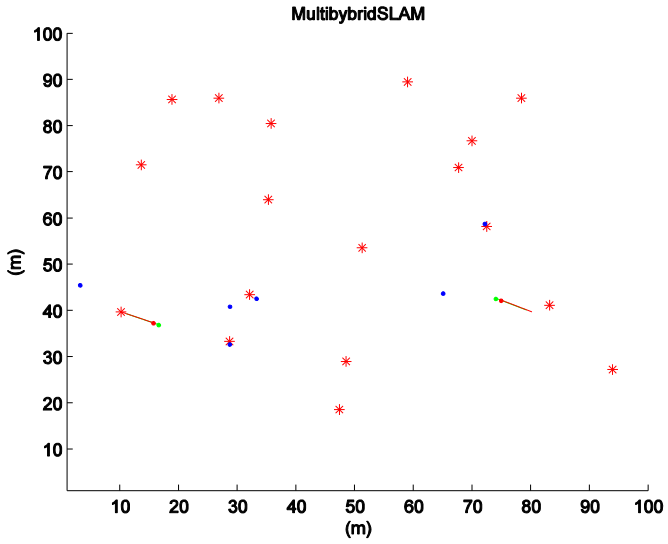


Figure 5.1(a): Multi-Hybrid SLAM simulation result at time = 20

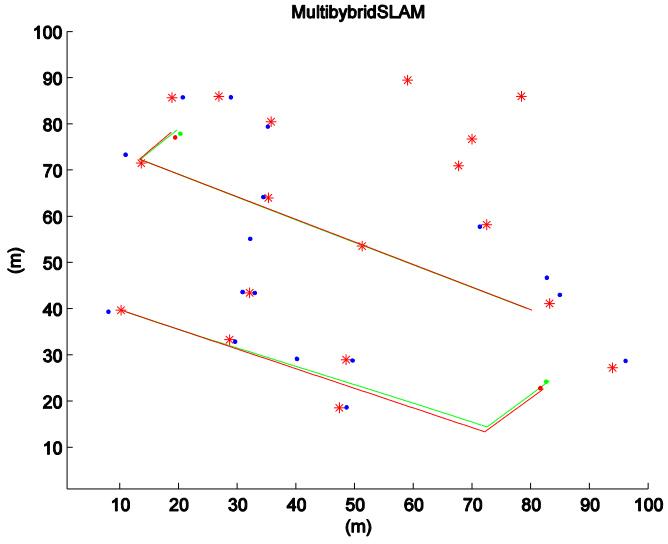


Figure 5.1(b): Multi-Hybrid SLAM simulation result at time = 100

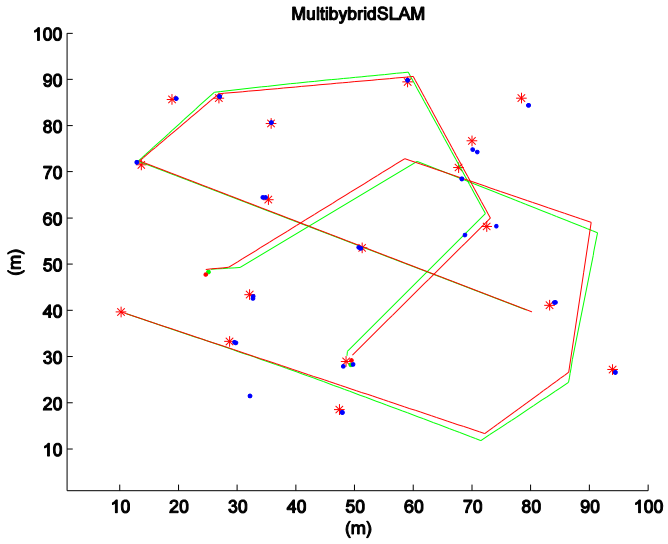


Figure 5.1(c): Multi-Hybrid SLAM simulation result at time = 200

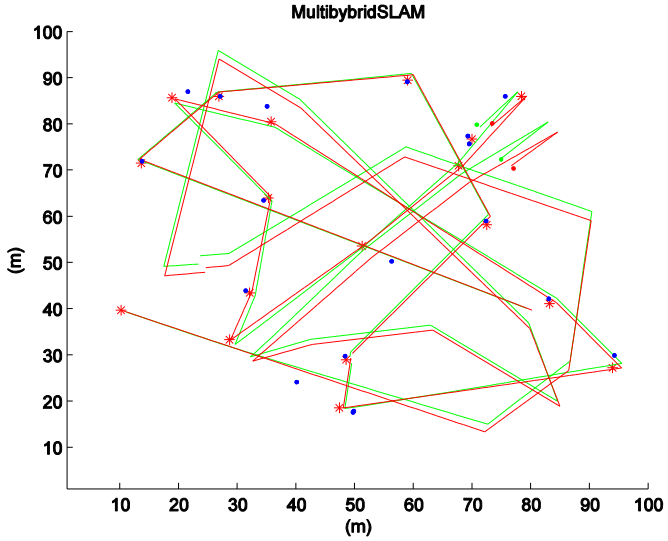


Figure 5.1(d): Multi-Hybrid SLAM simulation result at time = 500

From this result, we can see that the Hybrid SLAM algorithm could accomplish the task of team robot SLAM. In order to see its performance, it sets two experiments to compare its accuracy in robot pose estimations and landmark position estimations. In the following two sections, the experiences are set to compare with Multi-FastSLAM algorithm and single Hybrid-SLAM algorithm in the same environment.

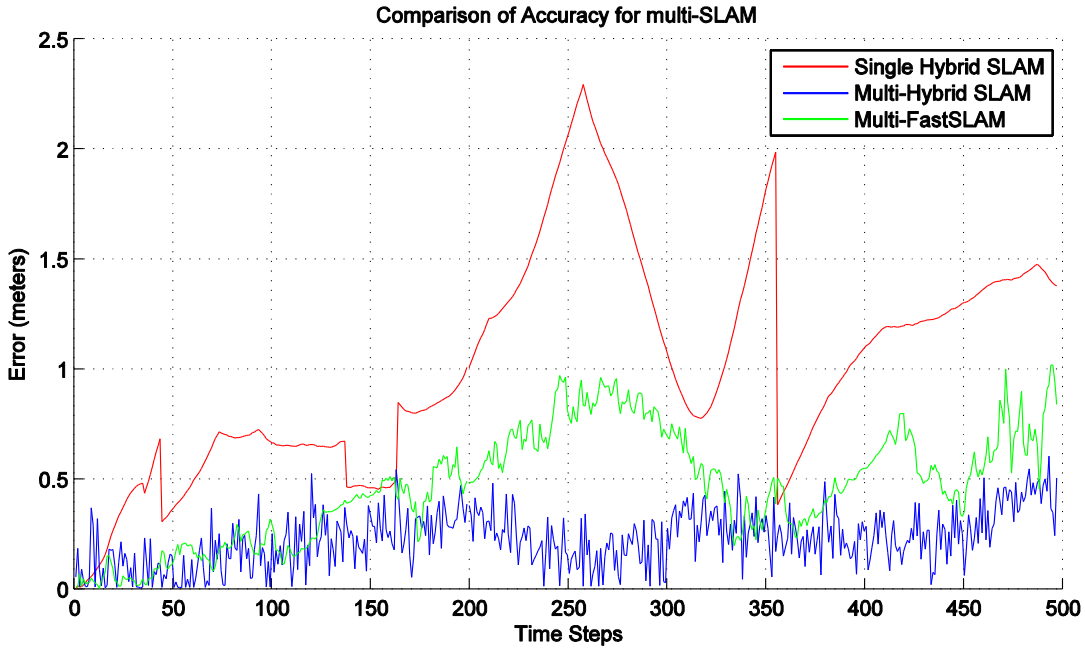


Figure 5.2: Experiment result on accuracy comparison of robot path

Figure 5.2 illustrates the simulation results of multi-Hybrid SLAM algorithm compared with single Hybrid SLAM algorithm and multi-FastSLAM algorithm in robot pose estimation. Red line indicates the distance error of single hybrid SLAM algorithm, green line is the estimation error of multi-FastSLAM algorithm, and the blue line is the estimation error of multi-Hybrid SLAM algorithm. The total time step is 500, and the number of particles is 50.

This figure clearly shows that multi-robot SLAM algorithms have smaller distance errors compared with single robot SLAM in robot path estimation. This is because particles' weights are not only updated in their own map, but also updated in other robot's map when two robots detected each other. Therefore, each particle could have an accurate weight to present the possible position of robot. Compared to the two multi-SLAM algorithms, the multi-Hybrid SLAM algorithm has a better result than multi-FastSLAM algorithm, with the error of path estimation within 0.5 meters. This is because the hybrid SLAM algorithm is superior to the FastSLAM algorithm, which has been clearly proved in Chapter 4.

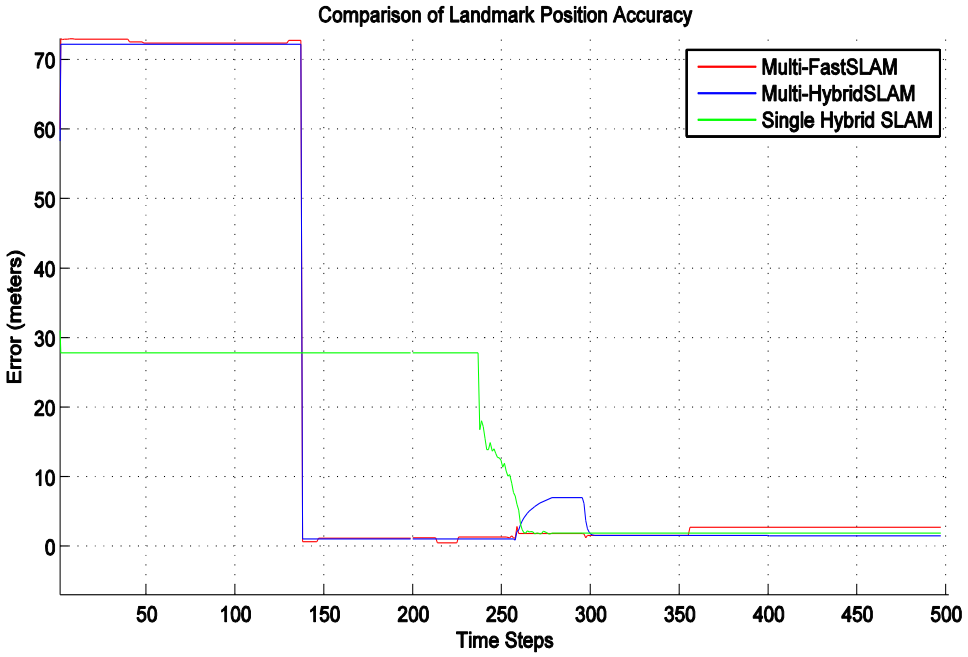


Figure 5.3: The performance of multi-Hybrid SLAM on landmark estimation against single Hybrid SLAM and multi-FastSLAM

Figure 5.3 compares the landmark estimation errors of multi-Hybrid SLAM algorithm with that of single Hybrid SLAM algorithm and multi-FastSLAM algorithm. The landmark is chosen by the first landmark that robots found in the environment. The blue line and red line indicates the errors of multi-Hybrid SLAM and multi-FastSLAM, respectively. The green line demonstrates the errors generated by single Hybrid SLAM. In this simulation, it records the errors made by robot 1.

The robot found the first landmark at its starting position. At the very beginning, errors are estimated by both multi-hybrid SLAM algorithm and multi-FastSLAM algorithm are very large. But the estimation error made by single Hybrid SLAM is relatively small. Then at time steps 120, robot 2 found the same landmark, and the estimation error made by multi-SLAM algorithms fall dramatically to a distance error of no more than 1 meter. But this error made by single Hybrid SLAM did not reduce until the 240th time step. This is because the robot is far away from this landmark before the 240th time step. Observation sensors could not detect this landmark. This

error reduced after 240th time step is because the robot moved, and it was close enough for sensors to detect this landmark. The final errors are very close. Therefore, these three SLAM algorithms are work successfully in estimating features of environment.

5.3 Discussion

This Chapter has illustrated a multi-Hybrid SLAM method to solve the multi-robot SLAM problem. The basic idea used in this thesis is to combine the same landmark's position from different robot map by using Kalman filter. Kalman filter is a mathematic tool to reduce estimation error by handling noise in the environment. However, Kalman filter has its limitation of linear problem. In other words, when combing landmarks, robots should stop and stand in environment and wait until the combination is finished. Because when robots move, the relationship between robot pose and the map is not linear. This is because of the motion model and observing model are not linear in robot SLAM. This algorithm works in Chapter 5 because the algorithm recorded those landmark positions to a new map out of robot's map. For every loop, robots will send their estimated landmark position to a new global map. Landmark combination is done on this new map.

This multi-hybrid SLAM gives two possible landmark positions, then merge them together to find a better estimation. It looks like here are two expected value (landmark positions), and then the algorithm sum them together. An optional idea is to use one robot's (robot A) landmark position and its covariance as the expected value. When another robot (robot B) detects this robot, robot B could use the landmark's position and covariance as observation information, and the distance between two robots could then be considered as the control. Jabobians is the observation value between robot B and the landmark. Then it could use the EKF to update the landmark information for robot B.

6 Conclusions and Future Work

This thesis was aimed at developing multi-robot SLAM algorithm through communication. The multi-Hybrid SLAM algorithm developed during this thesis showed an improvement on robot path estimation and landmark position estimation.

To develop a better filter to apply in multi-robot system, this thesis first compared current popular solutions to single robot SLAM in the literature level. It then selected three of the most popular solutions, which are EKF SLAM, FastSLAM, and the Hybrid methods, to test their performance in a simulation level. The simulated results show that the Hybrid method holds the smallest estimation errors in robot path estimation, compared to EKF SLAM and FastSLAM. This is because particle-based filter (FastSLAM and Hybrid SLAM) is more advanced in filtering out noise from true values. The current robot pose is estimated according to all its previous poses from the initial time in the Hybrid method, rather than its last time in FastSLAM. Therefore, robot path estimation error in the Hybrid method is smaller than the FastSLAM.

It also shows that, in the figure of landmark position estimation, the EKF method has the smallest error. This is probably because there are not sufficient particles, as the number of particle is set to 50 in this thesis. But the Hybrid method still performs better than the FastSLAM.

In single robot SLAM results, it also shows that robot path estimated by using the Hybrid method looks smoother than FastSLAM, and is not greatly influenced by odometric errors. This means that the Hybrid method can be applied in a wild range of automatic robots.

This thesis also expands this advanced Hybrid method to develop a multi-robot SLAM solution. From the simulation result, it shows that both robot path estimation and landmark position estimations are smaller than single robot estimation. This is

because in a robot team, the robot will acquire additional information from its teammate to narrow its SLAM posterior. The performance of multi-Hybrid method is the best as well, with the smallest estimation in robot path and the first observed landmark position. Therefore, we can say that the multi-robot SLAM algorithm has been developed in estimating robot pose and landmark positions in 2-D simulation level.

6.1 Contributions of this thesis

This thesis has presented a survey of current state-of-the-art filtering techniques used in mobile robot SLAM problem. It made main contributions to understand the popular SLAM algorithms, focusing on discussing the advantages and disadvantages of each approach. Considering of the strengths and weakness of different SLAM algorithms, it proposed a hybrid-SLAM algorithm used for team robots SLAM. In particular, the contributions of this thesis are,

1. A literature review of current works on mobile robot SLAM algorithms, with attention to those algorithms which can be used for multi-robots. (Chapter 2)

   These SLAM approaches (Kalman Filter, Expectation Maximization approach, Sparse Extended Information Filter, Particle Filter, and Sub Map Method) could be divided into two groups: Kalman Filter (KF)/Extended Kalman Filter (EKF) based SLAM, and Particle Filter (PF) based. The Kalman Filter based SLAM is a continued estimation of SLAM posterior. It assumes that robots current states are based on its previous pose, landmarks are relevant to current pose. However, Particle Filter based SLAM is a discrete estimation. Robot pose and positions of landmarks are independent to its previous time. The pros and cons of each method have been discussed in Table 2.1.

   Multi-robot SLAM strategies are mainly come from single robot SLAM algorithms. Most of the popular SLAM strategies have been extended for

multi-robot SLAM. Therefore, this thesis extended the Hybrid method SLAM algorithm to multi-robot SLAM.

2. This thesis describes standard EKF SLAM algorithm and PF/FastSLAM algorithms in detail. It explained math fundamentals of both algorithms. Then the simulations of both algorithms have been made to describe the results in 2-D environments. (Chapter 3)

3. A more advanced hybrid method was introduced by combing the strengths of EKF-based SLAM and PF-based SLAM algorithms (Chapter 4). The Hybrid method used FastSLAM to estimate robot poses and solve data associations. The EKF is used to record trajectory and to prevent the FastSLAM posterior over confidence. The result shows that the Hybrid method is better in estimations robot pose than EKF based SLAM, but has a equivalent level to FastSLAM.

4. In Chapter 5, the Hybrid SLAM algorithm was expanded into multi-robot SLAM algorithm. The idea of this algorithm is that each single robot perform single hybrid SLAM algorithm. The result of landmark positions will transfer to a central computer. The computer will merge positions from different robots. At last, this merged position will transfer back to each single robot. Results shows that the multi-hybrid algorithm successfully finished two robots SLAM task. The performance of multi-Hybrid SLAM algorithm is superb in robot pose estimations compared to the multi-FastSLAM algorithm and single Hybrid SLAM algorithm.

6.2 Future Work

Data association algorithms should be added into this multi-robot SLAM algorithm. In this thesis, it assumes data association problem has been exactly known. A further work needs to be done on applying the multi-Hybrid SLAM algorithm without knowing the data association problem. The basic idea to overcome this problem is based on it believes two landmark in the real world environment will not be too close.

Then for each observed landmark, it will set a small area. When another observed landmark position located in this area, it believes this is the observed landmark. If not, then it is a new landmark.

This multi-hybrid algorithm could be easily applied for three or more robots. Each member could send their landmarks to the global map. And this is not time consuming. The problem of this algorithm is how could we guarantee those landmarks we combined indicating the same landmark if the correspondence is not known. An idea to solve this problem is that we could combine those landmarks located very near to others. These may the same one on different robot maps. A further experiment will be set up to show this result.

It is also noticed that there is another way to update those landmarks that detected by other robots. When two robots (robot A and robot B) meet each other (distance and bearing between the two robots is *d*), both of them have observed one landmark θ. Then the landmark's position located in robot A's map could be considered as an expected position, and the landmark's position located in robot B's map could be considered as an observing position. Then we can calculate the combined mean value and the covariance for this landmark. The idea is that considering landmark position in robot A's map as the expected position. In EKF SLAM, this is the landmark position at time *t-1*. Then the observed distance *d* is considered as the control μ in EKF SLAM. Landmark position in robot B could be considered as the observed information at time *t* in EKF SLAM. Jacobian could use equation (4.15), where s is the location for robot B, Θ is the observed landmark for both robots. The advantage of this idea is that landmark position could be updated immediately when two robot meet. The information will not require transferred to a central computer. Robots could work independently in the environment without outside information.

Appendix

This is the Matlab code for multi-Hybrid SLAM simulation.

```
clear all;
NumberTimeStamps = 500;
MapDimension = [1,100;1,100];
NPARTICLES = 50;
MAX_RANGE= 20.0;
NEFFECTIVE=0.75*NPARTICLES;
pp=[10.238    13.66    26.891    59.054    72.513    48.561    47.421    93.955    83.234
     35.787    18.907    35.331    32.137    28.715    51.298    67.722    78.444    70.003;
    39.645    71.487    85.961    89.434    58.171    28.934    18.513    27.197    41.092
     80.461    85.671    63.961    43.408    33.276    53.539    70.908    85.961    76.697];
% landmarks positions
NumberLandmarks = size(pp,2);
%calculate the number of landmarks
figure(1); clf;
title('Introduce trajectory around landmarks');
v=[MapDimension(1,1) MapDimension(1,2) MapDimension(2,1) MapDimension(2,2)];
axis(v); hold on;
plot(pp(1,:),pp(2,:),'r*');
pin = 1; button = 0; fi = 0;
npoints = 0; dist = 0;
t=[80.23813.66    26.891    59.054    72.513    48.561    47.421    93.955    83.234    35.787
     18.907    35.331    32.137    28.715    51.298    67.722    78.444    70.003;
    39.645    71.487    85.961    89.434    58.171    28.934    18.513    27.197    41.092
     80.461    85.671    63.961    43.408    33.276    53.539    70.908    85.961    76.697];
% trajectory for robot 1
npoints=size(t,2);
for i=1:size(t,2)-1
     dist = dist + norm(t(:,npoints) - t(:,npoints-1));
     % calculate the length of robot 1's path
     npoints = npoints-1;
end
point = 2; dist2=0; incdist=dist/NumberTimeStamps;
tt(:,1)=t(:,1);
for i = 2:NumberTimeStamps
     tt(:,i) = tt(:,i-1) + incdist*((t(:,point)-t(:,point-1))/norm(t(:,point)-t(:,point-1)));
     vv(:,i-1)=tt(:,i)-tt(:,i-1);
     % vx,vy present velocities for robot 1. In this algorithm, it uses vx, vy to present the control.
     vx^2+vy^2=1.
```

```
        plot(tt(1,i),tt(2,i),'b');
        %plot the true robot path of robot 1
        dist2 = dist2 + incdist;
        if (dist2 + incdist) > norm(t(:,point)-t(:,point-1)) && abs((dist2 + incdist) - norm(t(:,point) -
t(:,point-1))) > abs(dist2-norm(t(:,point)-t(:,point-1)))
                point = point + 1; dist2 = 0;
        % this means it have finished plotting the whole path
        end
end
pin = 1; button = 0; fi = 0;
npoints = 0; dist = 0;
t2=[10.238      71.487      85.961      89.434      58.171      28.934      18.513      27.197      41.092
        80.461      85.671      63.961      43.408      33.276      53.539      70.908      85.961      76.697;
      39.645      13.66       26.891      59.054      72.513      48.561      47.421      93.955      83.234
        35.787      18.907      35.331      32.137      28.715      51.298      67.722      78.444      70.003];
% trajectory for robot 2
npoints=size(t2,2);
for i=1:size(t2,2)-1
        dist = dist + norm(t2(:,npoints) - t2(:,npoints-1));
        npoints = npoints-1;
end
point = 2; dist2=0; incdist=dist/NumberTimeStamps;
tt2(:,1)=t2(:,1);
for i = 2:NumberTimeStamps
        tt2(:,i)=tt2(:,i-1)+
incdist*((t2(:,point)-t2(:,point-1))/norm(t2(:,point)-t2(:,point-1)));
        vv2(:,i-1)=tt2(:,i)-tt2(:,i-1);
        plot(tt2(1,i),tt2(2,i),'b');
        dist2 = dist2 + incdist;
        if (dist2 + incdist) > norm(t2(:,point)-t2(:,point-1)) && abs((dist2+
incdist)-norm(t2(:,point)-t2(:,point-1))) > abs(dist2-norm(t2(:,point)-t2(:,point-1)))
                point = point + 1; dist2 = 0;
        end
end
[velocity,del_theta,theta,theta_r]=caculate_velocity(vv,NumberTimeStamps);
% calculate velocity for robot1
[velocity2,del_theta2,theta2,theta_r2]=caculate_velocity(vv2,NumberTimeStamps);
%calculate velocity for robot2
xtrue=[tt;theta];
% true path for robot 1
xtrue2=[tt2;theta2];
% true path for robot 2
[particle_xv,particle_w,particle]=initialize_particle(xtrue(:,1), NPARTICLES);
% initialization for robot 1
```

```
[particle_xv2,particle_w2,particle2]=initialize_particle(xtrue2(:,1),NPARTICLES);
%initialization for robot 2
r = [(15)^2 0; 0 10*pi/18^2];       % initial uncertainty for landmarks
rg = (3.0*pi/180)^2;                % control error in distance
rv = (0.05)^2;                      % control error in angle
R= [2^2 0; 0 pi/18^2];              % observing error
ftag= 1:NumberLandmarks;
% ftag is used to indicate robot 1's orders of landmarks
ftag2= 1:NumberLandmarks;
% ftag is used to indicate robot 2's orders of landmarks
g(NumberTimeStamps)=0;
Vn=0; Gn=0;
Vn2=0; Gn2=0;
plandmarks=pp;
da_table= zeros(1,size(pp,2));
% data association table for robot 1
da_table2= zeros(1,size(pp,2));
% data association table for robot 2
for i=1:NumberTimeStamps-1
    [Vn,Gn,theta_Gn]= add_control_noise(rv,rg,NPARTICLES);
    % add control noise, the input is control noise parameter for distance and headings, the output is a set Gaussian noise formulated by (0,rv^2) and (0. rg^2)
    [Vn2,Gn2,theta_Gn2]= add_control_noise(rv,rg,NPARTICLES);
    % this is the noise for robot 2
   [particle_xv]=predict_robot_pose(particle_xv,velocity(i),del_theta(i),Vn,Gn,NPARTICLES);
    % robot 1's predicated, the predicted pose is calculated by adding robot control and control noise
   poses[particle_xv2]=predict_robot_pose(particle_xv2,velocity2(i),del_theta2(i),Vn2,Gn2,NPARTICLES);
    % robot 2's predicated, the predicted pose is calculated by adding robot control and control noise
    [z,ftag_visible]= get_observations(xtrue(:,i), pp, ftag, MAX_RANGE);
    % sensing observed landmarks. the input is robot 1's true pose, all landmark positions, ftag and max sensing range; the output is ftag for observed landmark for robot 1
    [z2,ftag_visible2]= get_observations(xtrue2(:,i), pp, ftag2, MAX_RANGE);
    % sensing observed landmarks. the input is robot 2's true pose, all landmark positions, ftag and max sensing range; the output is ftag for observed landmark for robot 2
    z= add_observation_noise(z,R);
    % add observing noise for robot 1
    z2= add_observation_noise(z2,R);
    % add observing noise for robot 1
    Nf= size(particle(1).lm,2);
    %numbers for observed landmarks of robot 1, this is used for add new landmarks to "particle(1).lm,2"
```

```
    Nf2=size(particle2(1).lm,2);
    %numbers for observed landmarks of robot 2, this is used for add new landmarks to
"particle2(1).lm,2"
   [zf,idf,zn,da_table]= data_associate_known(z, ftag_visible, da_table,Nf);
    %store new and old landmarks in zn and zf for robot 1
   [zf2,idf2,zn2,da_table2]= data_associate_known(z2, ftag_visible2, da_table2,Nf2);
    %store new and old landmarks in zn and zf for robot 2
     for j=1:NPARTICLES
         if ~isempty(zf)
         % if robot 1 finds old landmarks, landmark information should be updated and particle's
weight should be calculated
              [w,zp]= compute_weight(particle_xv(:,j),particle(j),zf,idf,R);
            %compute the weight
             particle_w(j)= particle_w(j)*w;
            %normalize particle's weight, if not weight will be too small to calculate
             particle(j)= landmark_update(particle_xv(:,j), particle(j),zf, idf, R);
            %update old landmarks
         end
          if~isempty(zn)
          % if robot 1 finds new landmarks, landmark information should be added into the
particle
              [particle_xv(:,j),particle(j)]=add_landmark(particle_xv(:,j),particle(j),zn,r,NPARTI
CLES);
                  % add new landmark to tables
          end
     end
  sum_weight = sum(particle_w);
  if sum_weight~=0
      particle_w=particle_w/sum_weight;
  else
%compute the weight for each particle for robot 1
      for j=1:NPARTICLES
         particle_w(j)=1/NPARTICLES;
      end
% for the next time step, particles' weight should be initialized; this is different from FastSLAM,
because in Hybrid SLAM, all particles' positions will be set on the weighted mean positions
  end
  nn=size(particle(1).lm,2);
  % calculate the weighted mean as well as the covariance by using hybrid method
  lmx=zeros(1,nn);lmy=zeros(1,nn);
  for j=1:NPARTICLES
      wx(j)=particle_xv(1,j)*particle_w(j);     % weighted mean for x variable of robot 1
      wy(j)=particle_xv(2,j)*particle_w(j);     % weighted mean for y variable of robot 1
      wz(j)=particle_xv(3,j)*particle_w(j);     % weighted mean for heading variable of robot 1
```

```
        lm(j).x(1,1:nn)=particle(j).lm(1,1:nn)*particle_w(j);
        % weighted landmark positions in x coordinate
        lm(j).x(2,1:nn)=particle(j).lm(2,1:nn)*particle_w(j);
        % weighted landmark positions in y coordinate
        lmx(1:nn)=lmx(1:nn)+lm(j).x(1,1:nn);
        lmy(1:nn)=lmy(1:nn)+lm(j).x(2,1:nn);
    end
    real(1,i)=sum(wx);
    real(2,i)=sum(wy);
    real(3,i)=sum(wz);
    for j=1:NPARICLES
        cov1_x(1:nn)=particle_w(j)*(cov1_x(1:nn)+( lm(j).x(1,1:nn)-lmx(1:nn))^2);
        % covariance for robot 1in xx
        cov1_y(1:nn)=particle_w(j)*(cov1_y(1:nn)+( lm(j).x(2,1:nn)-lmx(1:nn))^2);
        % covariance for robot 1 in yy
    end
    for j=1:NPARTICLES
        particle_xv(:,j)=real(:,i);
        particle(j).lm(:,1:nn) = [lmx(1:nn);lmy(1:nn)];
    end
        for j=1:NPARTICLES              % done for robot 2
            if ~isempty(zf2)
                    [w2,zp2]= compute_weight(particle_xv2(:,j), particle2(j), zf2, idf2, R);
                      particle_w2(j)= particle_w2(j)*w2;
                      particle2(j)= landmark_update(particle_xv2(:,j), particle2(j), zf2, idf2, R);
            end
            if~isempty(zn2)

[particle_xv2(:,j),particle2(j)]=add_landmark(particle_xv2(:,j),particle2(j),zn2,r,NPARTICLES);
            end
        end
    sum_weight2 = sum(particle_w2);
    if sum_weight2~=0
        particle_w2=particle_w2/sum_weight2;
    else
        for j=1:NPARTICLES
           particle_w2(j)=1/NPARTICLES;
        end
    end
    nn2=size(particle2(1).lm,2);
    lmx2 = zeros(1,nn2);lmy2 = zeros(1,nn2);
    for j=1:NPARTICLES
        wx2(j)=particle_xv2(1,j)*particle_w2(j);
        wy2(j)=particle_xv2(2,j)*particle_w2(j);
```

```
        wz2(j)=particle_xv2(3,j)*particle_w2(j);
        lm2(j).x(1,1:nn2)=particle2(j).lm(1,1:nn2)*particle_w2(j);
        lm2(j).x(2,1:nn2)=particle2(j).lm(2,1:nn2)*particle_w2(j);
        lmx2(1:nn2)=lmx2(1:nn2)+lm2(j).x(1,1:nn2);
        lmy2(1:nn2)=lmy2(1:nn2)+lm2(j).x(2,1:nn2);
   end
   real2(1,i)=sum(wx2);
   real2(2,i)=sum(wy2);
   real2(3,i)=sum(wz2);
   for j=1:NPARICLES
        cov2_x(1:nn)=particle_w(j)*(cov2_x(1:nn)+( lm2(j).x(1,1:nn)-lmx2(1:nn))^2);
        cov2_y(1:nn)=particle_w(j)*(cov2_y(1:nn)+( lm2(j).x(2,1:nn)-lmy2(1:nn))^2);
   end
   for j=1:NPARTICLES
        particle_xv2(:,j)=real2(:,i);
        particle2(j).lm(:,1:nn2) = [lmx2(1:nn2);lmy2(1:nn2)];
   end
  [particle,da_table]=recieve_information(da_table,da_table2,particle,particle2,pp,NPARTICLES);
[particle2,da_table2]=recieve_information(da_table2,da_table,particle2,particle,pp,NPARTICLE);
  %merge the map
      clf; hold on; title('MultibybridSLAM');                    % plotting the results
      plot(pp(1,:),pp(2,:),'r*');
      v=[MapDimension(1,1) MapDimension(1,2)
         MapDimension(2,1) MapDimension(2,2)];
      axis(v);
      plot(real(1,1:i),real(2,1:i),'g');
      plot(real(1,i),real(2,i),'g.');
      plot(real2(1,1:i),real2(2,1:i),'g');
      plot(real2(1,i),real2(2,i),'g.');
      plot(xtrue(1,i),xtrue(2,i),'r.');
      plot(xtrue(1,1:i),xtrue(2,1:i),'r');
      plot(xtrue2(1,i),xtrue2(2,i),'r.');
      plot(xtrue2(1,1:i),xtrue2(2,1:i),'r');
      for k=1:nn
          plot(lmx(1,k),lmy(1,k),'b.');
      end
      bbb(:,1:nn,i)=[lmx(1:nn);lmy(1:nn)];
      for k=1:nn2
          plot(lmx2(1,k),lmy2(1,k),'b.');
      end
      drawnow; hold off;
end
function [velocity,del_theta,theta,del_theta_r] =caculate_velocity(vv,NumberTimeStamps)
% this function is to calculate robot control over all time, oututs are velocities in distance and
```

```
angles.
for i=1:NumberTimeStamps-1
    velocity(i)=norm([vv(1,i);vv(2,i)]);
    theta(i+1)=atan2(vv(2,i),vv(1,i));
    del_theta(i)=theta(i+1)-theta(i);
    del_theta_r(i)=theta(i)*180/pi;
end
function [particle_xv,particle_w,particle]=initialize_particle(xtrue,np)
% this is initialization function, particle weights are 1/numbers of particles, the initial landmark for each particle is empty
for i=1:np
    particle_w(i)= 1/np;
    particle_xv(:,i)= xtrue;
    particle(i).lm= [];
    particle(i).lp= [];
end
function [Vn,Gn,theta_Gn]= add_control_noise(V,G,np)
% formulating control noise
for i=1:np
    Vn(i)=randn(1)*V;
    Gn(i)=randn(1)*G;
    theta_Gn(i)=180/pi*Gn(i);    % in degrees
end
function particle_xv=predict_robot_pose(particle_xv,velocity,del_theta,Vn,Gn,np)
% this is robot motion model, knowing robot pose at last time, and guess robot pose at current time
 for j=1:np
     particle_xv(:,j)=[particle_xv(1,j)+(velocity+Vn(j))*cos(Gn(j)+del_theta+particle_xv(3,j));
     particle_xv(2,j)+(velocity+Vn(j))*sin(Gn(j)+del_theta+particle_xv(3,j));
     particle_xv(3,j)+del_theta+Gn(j)];
 end
function [z,idf]= get_observations(x, lm, idf, rmax)
% get robot observed landmarks, inputs are robot current pose, all landmarks, and the max sensing range, outputs are the distance and bearings between robot pose and observed landmarks, and the indications of landmarks in landmark table
[lm,idf]= get_visible_landmarks(x,lm,idf,rmax);
z= compute_range_bearing(x,lm);
function [lm,idf]= get_visible_landmarks(x,lm,idf,rmax)
dx= lm(1,:) - x(1);      % distance in x-coordinate
dy= lm(2,:) - x(2);      % distance in y-coordinate
phi= x(3);                % angels
ii= find((dx.^2 + dy.^2) < rmax.^2);          % find which landmark is oberved
lm= lm(:,ii);                 %store the result
idf= idf(ii);                 % store the indications
```

```
function z= compute_range_bearing(x,lm)
dx= lm(1,:) - x(1);
dy= lm(2,:) - x(2);
phi= x(3);
z= [sqrt(dx.^2 + dy.^2);                    % the distances and angels for observed landmarks
    pi_to_pi(atan2(dy,dx) - phi)];      % limited the angle between –pi to pi
function z= add_observation_noise(z,R)
% for all observed landmark positions, it adds a noise to them
len= size(z,2);
    if len > 0
        z(1,:)= z(1,:) + randn(1,len)*sqrt(R(1,1));     %randomly add a noise
        z(2,:)= z(2,:) + randn(1,len)*sqrt(R(2,2));
end
function [zf,idf,zn,table]= data_associate_known(z, idz, table,Nf)
% find which landmarks are observed and which are not in landmark table
zf= []; zn= [];
%zf presents observed landmark positions; zn presents new landmark positions
idf= []; idn= [];
% the indications of observed (idf) and new(idn) landmarks
for i=1:length(idz)
    ii= idz(i);
    if table(ii) == 0
    % when a landmark is found, then the data association table will change for 0 to 1, which mean it has been corresponded.
        zn= [zn z(:,i)];
        idn= [idn ii];
        table(ii)=1;
    else
        zf= [zf z(:,i)];
        idf= [idf table(ii)];
    end
end
table(idn)= Nf+ (1:size(zn,2));             % add new features to data association table
function [w,zp]= compute_weight(particle_xv, particle, zf, idf, R);
% inputs are particle positions, updates particle weights according to those observed landmarks
[zp,Hv,Hf,Sf,d]= compute_jacobians (particle_xv,particle, idf, R);
% this parts could reference Fastslam tutorials by Montement    on his phd thesis.
v= zf-zp;
v(2,:)= pi_to_pi(v(2,:));
w= 1;
for i=1:size(zf,2)
    S= Sf(:,:,i);
    den= sqrt(2*pi*det(S));
    num= exp(-0.5 * v(:,i)' * inv(S) * v(:,i));
```

```
        w = w*num/den;
end
function [zp,Hv,Hf,Sf,dd]= compute_jacobians(particle_xv, particle,idf, R)
xv= particle_xv;
xf= particle.lm(:,idf);
Pf= particle.lp(:,:,idf);
for i=1:length(idf)
        dx= xf(1,i)-xv(1);        % distances in x
        dy= xf(2,i)-xv(2);        % distances in y
        d2= dx.^2 + dy.^2;      % distances in angle
        d= sqrt(d2);                % distances
        dd(:,i)=d;
        zp(:,i)= [d; pi_to_pi(atan2(dy,dx) - xv(3))];
        Hv(:,:,i)= [-dx/d,    -dy/d,      0;                    % jacobians
                      dy/d2, -dx/d2, -1];
        Hf(:,:,i)= [ dx/d,      dy/d;
                    -dy/d2,    dx/d2];
        Sf(:,:,i)= Hf(:,:,i) * Pf(:,:,i) * Hf(:,:,i)' + R;           % observe noise
end
function particle= landmark_update(particle_xv, particle, zf, idf, R)
        lm=particle.lm(:,idf);
        lp=particle.lp(:,:,idf);
        [zp,Hv,Hf,Sf]= compute_jacobians(particle_xv,particle, idf, R);
        v= zf-zp;
        for i=1:length(idf)
        vi= v(:,i);
        Hfi= Hf(:,:,i);
        Pfi= lp(:,:,i);
        xfi= lm(:,i);
        [lm(:,i), lp(:,:,i)]= KF_cholesky_update(xfi,Pfi, vi,R,Hfi);
end
particle.lm(:,idf)= lm;      % store landmark positions
particle.lp(:,:,idf)= lp;      % store covariance for landmarks
function [particle_xv,particle]=add_landmark(particle_xv,particle,z,R,np)
% inputs are new landmarks and old particle information, outputs are new landmark information,
this information adds new landmark position and covariances.
        lenz= size(z,2);
        lm= zeros(2,lenz);
        lp= zeros(2,2,lenz);
        xv= particle_xv;
for i=1:lenz
        r= z(1,i); b= z(2,i);
        s= sin(xv(3)+b);
        c= cos(xv(3)+b);
```

```
        lm(:,i)= [xv(1) + r*c;
                  xv(2) + r*s];
        Gz= [c r*s;
             s r*c];
        lp(:,:,i)= Gz*R*Gz';
end
lenx= size(particle.lm,2);
ii= (lenx+1):(lenx+lenz);
        particle.lm(:,ii)= lm;
particle.lp(:,:,ii)= lp;
function
[recieve,recieve_table]=recieve_information(recieve_table,send_table,recieve,send,pp,np)
% this is function transferring information to the central computer to merge two robot's individual
map
lm=[];lp=[];
% initial summed map is empty
send_lm=[];send_lp=[];
recieve_lm=[];recieve_lp=[];
matrix=[]; var=[];
for i=1:length(pp)
        len=size(recieve(1).lm,2);
        if recieve_table(i)==0 & send_table(i)~=0
        % if the landmark is never merged or never found, just add a new landmark
              for j=1:np
                    n = send_table(i);
                    lm = send(j).lm(:,n);
                    lp = send(j).lp(:,:,n);
                    recieve(j).lm(:,len+1)=lm;
                    recieve(j).lp(:,:,len+1)=lp;
              end
              recieve_table(i)=len+1;
        end
              if recieve_table(i)~=0 & send_table(i)~=0
         % for those observed landmarks, sum them togethor
                    for j=1:np
                          n = send_table(i);
                          m = recieve_table(i);
                          send_lm = send(j).lm(:,n);
                          send_lp = send(j).lp(:,:,n);
                          recieve_lm = recieve(j).lm(:,m);
                          recieve_lp = recieve(j).lp(:,:,m);
                          var = send_lm'*(recieve_lp.^2/(send_lp.^2+recieve_lp.^2))+
                          recieve_lm'*(send_lp.^2/(send_lp.^2+recieve_lp.^2));
                          send(j).lm(:,n)=var';
```

```
                    recieve(j).lm(:,m)=var';

matrix(1,1)=send_lp(1,1).^2.*recieve_lp(1,1).^2/(send_lp(1,1).^2+recieve_lp(1,1).^2);
matrix(1,2)=send_lp(1,2).^2.*recieve_lp(1,2).^2/(send_lp(1,2).^2+recieve_lp(1,2).^2);
matrix(2,1)=send_lp(2,1).^2.*recieve_lp(2,1).^2/(send_lp(2,1).^2+recieve_lp(2,1).^2);
matrix(2,2)=send_lp(2,2).^2.*recieve_lp(2,2).^2/(send_lp(2,2).^2+recieve_lp(2,2).^2);
                    send(j).lp(1,1)=sqrt(matrix(1,1));
                    send(j).lp(1,2)=sqrt(matrix(1,2));
                    send(j).lp(2,1)=sqrt(matrix(2,1));
                    send(j).lp(2,2)=sqrt(matrix(2,2));
                    recieve(j).lp(:,:,m) = send(j).lp(:,:,n);
            end
        end
end
```

References

Andreas, Surmann, H., Lingemann, K., Hertzberg, J. & Thrun, S. (2004), 6D SLAM with an application in autonomous mine mapping, *in* 'The IEEE International Conference on Robotics and Automation', pp. 1998-2003.

Bailey, T., Nieto, J., Guivant, J., Stevens, M. & Nebot, E.(2006), Consistency of the EKF-SLAM algorithm, *in* 'IEEE/RSJ International Conference on Intelligent Robots and Systems', pp. 3562-3568.

Bailey, T., Nieto, J. & Nebot E. (2006), Consistency of the FastSLAM algorithm, *in* 'IEEE International Conference on Robotics and Automation, 2006.', pp. 424 – 429.

Brooks, A. & Bailey, T. (2009), HybridSLAM: Combining FastSLAM and EKF-SLAM for reliable mapping, *in* 'Algorithm Foundation of Robotics VIII', Springer Berlin/ Heidelberg.

Chang, H. J., Lee, C. S. G., Hu, Y. C. & Lu, Y. H. (2007), Multi-robot SLAM with topological/ metric maps, *in* 'IEEE/RSJ International Conference on Intelligent Robots and Systems', pp. 1467 – 1472.

Cheein, F. A., Lopez, N., Soria, C., di Sciascio, F., Pereira, F. L. & Carelli, R. (2010), 'SLAM algorithm applied to robotics assistance for navigation in unknown environments', *Journal of NeuroEngineering and Rehabilitation* 7, 10-15.

Csorba, M. (1997), Simultaneous localization and map building, PhD thesis, University of Oxford.

Dellaert, F., Fox, D., Rurgard, W. & Thrun, S. (1999), Monte Carlo localization for mobile robots, *in* 'IEEE International Conference on Robotics and Automation (ICRA99)', PP.26-34.

Dellaert, F., Seitz, S. M., Thorpe, C. E. & Thrun, S. (2003), 'EM, MCMC, and chain flipping for structure from motion with unknown correspondence', *Machine Learning* 50 (1-2), 45-71.

Dissanayake, G., Durrant-Whyte, H. F. & Bailey, T. (2000), A computationally efficient solution to the simultaneous localization and map building (SLAM) problem, *in* 'International Conference on Robotics and Automation', pp. 1009-1014.

Dissanayak, G., Williams, S. B., Durrant-Whyte, H. & Bailey, T. (2002), 'Map management for efficient simultaneous localization and mapping (SLAM)', *Auton. Robots* 12 (3), 267-286.

Doucet, A., Freitas, N., Murphy, K. & Russell, S. (2000), Rao-Blackwellised particle filtering for dynamic Bayesian networks, *in* 'in The 16th Annual Conference on Uncertainty in Artificial Intelligence', Morgan Kaufmann Publishers, pp. 176-183.

Eustice, R. & Ma, W. H. (2005), Sparse extended information filters: Insights into scarification, *in* ' IEEE/RSJ International Conference on Intelligent Robots and Systems', pp. 641-648.

Fox, D., Thrun, S., Dellaert, F. & Burgard, W. (2000), Particle filters for mobile robot localization, *in* A. Doucet, N. de Freitas & N. Gordon, eds, 'Sequential Monte Carlo Methods in Prctice', Springer Verlag, New York.

Guivant, J., Nebot, E. & Baiker, S. (2000), 'Localization and map building using laser range sensors in outdoor applications', *Journal of Robotic Systems* 17 (10), 565-583.

Howard, A. (2006), 'Multi-robot simultaneous localization and mapping using particle filters', *International Journal Robots Research* 25 (12), 1243-1256.

Huang, S. & Dissanayake, G. (2007), 'Convergence and consistency analysis for extended Kalman filter based SLAM', *Trans. Rob*. 23(5), 1036-1049.

Kummerle, R., Steder, B., Dornhege, C., Ruhnke, M., Grisetti, G., Stachniss, C. & Kleiner, A. (2009), 'On measuring the accuracy of SLAM algorithms', *Journal of Autonomous Robots* 27(4), 387-407.

Liu, J. S. (2008), *Monte Carlo Strategies in Scientific Computing*, corrected edn, Springer.

Liu, Y. & Thrun, S. (2002), Results for outdoor-SLAM using sparse extended information filters, *in* 'IEEE International Conference on Robotics and Automation ICRA', pp. 1227-1233.

Marjovi, A., Nunes, J. a. G., Marques, L. & de Almeida, A. (2009), Multi-robot exploration and fire searching, *in* 'IEEE/RSJ International Conference on Intelligent Robots and Systems', IROS'09, IEEE Press, Piscataway, NJ, USA, pp. 1929-1934.

Martijn, R. & Andreas, B. (2007), 'Multi-robot exploration under the constraints of wireless networking', *Control Engineering Practice* 15(4), 435-445.

Montemerlo, M., Thrun, S., Koller, D. & Wegbreit, B. (2002), FastSLAM: A factored solution to the simultaneous localization and mapping problem, *in* 'The AAAI National Conference on Artificial Intelligence', AAAI, Edmonton, Canada, pp. 593-598.

Paskin, M. A. (2003), Thin junction tree filters for simultaneous localization and mapping, *in* 'The 18th International Joint Conference on Artificial Intelligence', IJCAI'03, Morgan Kaufmann Publishers Inc., pp. 1157-1164.

Roman, C. N. & Singh, H. (2005), Improved vehicle based multibeam bathymetry

bathymetry using sub-maps and SLAM, *in* 'IEEE/RSJ International Conference on Intelligent Robots and Systems', pp. 3662-3669.

Ruhnke, M., Kmmerle, R., Grisetti, G. & Burgard, W. (2011), Highly accurate maximum likelihood laser mapping by jointly optimizing laser points and robot poses, *in* 'Proc. of the IEEE Int. Conf. on Robotics Automation (ICRA)', pp. 2812-2817.

Seib, V., Gossow, D., Vetter, S. & Paulus, D. (2011), Hierarchical multi-robot coordination, *in* 'RoboCup 2010: Robot Soccer World Cup XIV', Vol. 6556, Springer Berlin/Heidelberg, pp. 314-323.

Smith, R. C. & Cheeseman, P. (1986), 'On the representation and estimation of spatial uncertainty', *International Journal Robotics Research 5* (4), 56-68.

Smith, R., Self, M. & Cheeseman, P. (1990), 'Estimating uncertain spatial relationships in robotics', *Autonomous Robot Vehicles* 8, 167-193.

Stachniss, C. & Burgard, W. (2004), Exploration with active loop-closing for FastSLAM, *in* 'The IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', pp. 1505-1510.

Tesli, L., Krjanc, I. & Klanar, G. (2011), 'EKF-based localization of a wheeled mobile robot in structured environments', *Journal of Intelligent and Robotic Systems* 62 (2),187-203.

Thrun, S. (2002), Robotic mapping: A survey, *in* 'Exploring Artificial Intelligence in the New Millenium', Morgan Kaufmann.

Thrun, S., Burgard, W. & Fox, D. (2005), *Probabilistic Robotics*, MIT Press.

Thrun, S., Burgard, W. & Fox, D., Hexmoor, H. & Mataric, M. (1998), A probabilistic approach to concurrent mapping and localization for mobile robots, *in* 'Machine

Learning', Vol. 5, pp. 29-53.

Thrun, S., Fox, D., Burgard, W. & Dellaert, F. (2001), 'Robust Monte Carlo localization for mobile robots', *Artificial Intelligence Journal* 128 (1-2).

Thrun, S. & Liu, Y. (2003), Multi-robot slam with sparse extended information filters, *in* 'Robotics Research', Springer, pp. 254-266.

Thrun, S., Martin, C., Liu, Y., Dirk, H., Emerty- Montemerlo, R., Chakrabarti, D. & Burgard, W. (2004), 'A real-time expectation-maximization algorithm for acquiring multiplanar maps of indoor environemtns with mobile robots.', *IEEE Transactions on Robotics* 20(3), 433-443.

Walter, M. R., Eustice, R. M. & Leonard, J. J. (2007), 'Exactly sparse extended information filters for feature-based slam', *International Journal Robotics Research* 26 (4), 335-359.

Wan. E. A. & Merwe, R. V. D. (2000), The unscented Kalman filter for nonlinear estimation, *in* 'Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000', pp.153-158.

Wang, Z., Huang, S. & Dissanayake, G. (2007), 'Multi-robot simultaneous localization and mapping using D-SLAM framework', *The Third International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP).*

Wu, M., Huang, F., Wang, L. & Sun, J. (2009), Cooperative multi-robot monocular-slam using salient landmarks, *in* '2009 International Asia Conference on Informatics in Control, Automation and Robotics', CAR'09, IEEE Computer Society, Washington, DC, USA, pp. 151-155.

Zhou, X. S. & Roumeliotis, S. I. (2006), Multi-robot slam with unknown initial

correspondence: The robot rendezvous case, *in* ‘The IEEE International Conference on Intelligent Robots and Systems’, pp. 1785-1792.