MULTI ROBOT FASTSLAM

DILHAN BALAGE

Multi Robot FastSLAM

by

© Dilhan Balage, B.Sc.Eng(Hons)

A thesis submitted to the
School of Graduate Studies
in partial fulfillment of the
requirements for the degree of

Master of Engineering

Faculty of Engineering and Applied Science
Memorial University of Newfoundland
May, 2010
St. John's, Newfoundland, Canada

Abstract

Robotic mapping has been an active research area in robotics for last two decades. An accurate map is a mandatory requirement for a robot to work autonomously. In addition the robot requires to know its position with respect to a given map and this is solved through robot localization. The problem of solving both map building and robot localization is addressed by simultaneous localization and mapping (SLAM). A large volume of literature is available to solve the SLAM problem using a single robot. A robot will take a series of sensor readings about an unexplored area and then continues to build the map while knowing its position reference to partially built map. However, when an area becomes larger multi-robot SLAM is more efficient and also has the advantage of sharing the computational burden among several robots. Solving SLAM problem using multiple robot is important when there is large terrain to map and perhaps it will beyond the capability of single robot. Even if it is within the capability of single robot, such a deployment will not be cost and time effective. Therefore this research focuses on developing a multi-robot SLAM filter based on FastSLAM algorithm.

Single Pioneer 3AT robot was deployed to collect odometry and sensor readings. Grid based fastSLAM algorithm is implemented on MATLAB program code for off-line processing and successfully generated the map of the environment. The data set obtained from single robot was divided into two data sets and they were treated as if they were obtained from two different robots. Single robot grid based fastSLAM algorithm was applied to both of the data sets and obtained two maps. Two maps were merged using Hough transform based map merging technique. Maps obtained from single robot SLAM and multi-robot SLAM is compared and multi-robot SLAM algorithm provides maps as same accuracy as single robot SLAM.

Acknowledgments

I would like to thank my supervisors Dr. George Mann and Dr. Faisal Khan for their support and continuing encouragement, without which I could not have completed this programme. They have encouraged me throughout my candidature, offered me words of wisdom when needed, and allowed me the freedom to explore new areas. Also I would like to thank the INCO innovation fund for providing financial assistance for this research.

I would thank every one of my colleagues at the Intelligent Systems Laboratory (ISLAB): Dilan, Awantha, Gayan, Jeffery, Farid, Nusrat and Dave, from whom I learnt so much and for their friendship during my stay.

I appreciate all the care and unwavering support from my parents and sisters during the long years that I have been away from home. At last, I would like to thank my wife for love, respect and support. Without her company, life would have never been so exciting; to whom I dedicate this thesis.

Contents

List of Tables

List of Figures

List of Symbols

Chapter 2

m : Set of all landmarks

m_i : Cartesian coordinates of i^{th} landmark in the world coordinates

M : Number of particles

N : Number of landmarks

P^T : Transpose of vector P or matrix P

$u(k)$: Control vector, applied at time $k-1$ to drive the robot to a state $x_r(k)$ from state $x_r(k-1)$ at time k

$u_{1:k}$: History of control inputs from time step 1 to k

$x(k)$: State vector which contains both pose of the robot and map at time k

$x_r(k)$: Pose of the robot at time k

$x_r(1:k)$: Complete pose history of the robot

$z(k)$: Landmarks in the observation at time k

$z_i(k)$: Observation of i^{th} landmark at time k

$z_{1:k}$: History of sensor data measurements from time step 1 to k

Chapter 3

$E_{dist}(\omega, T)$: Distance function between two point sets

m : Map

ω : Rotation

N_{eff} : Number of effective particles

T : Translation

T_x : Translation along x-direction

T_y : Translation along y-direction

$w_t^{(i)}$: Importance weight of i^{th} particle at time t

$\tilde{w}^{(i)}$: Normalized importance weight of particle i

η : Normalization factor

$\mathcal{G}^{(i)}$: Topological map correspond to i^{th} particle

Chapter 4

M : Map

$HT_{\mathcal{M}}$: Hough Transform of map M

$HS_{\mathcal{M}}$: Hough Spectrum of map M

$CC_{\mathcal{M}_1\mathcal{M}_2}$: Circular cross correlation of Hough spectrums of maps M_1 and M_2

$SX_{\mathcal{M}}$: X-spectrum of map M

$SY_{\mathcal{M}}$: Y-spectrum of map M

$CCX_{\mathcal{M}_1\mathcal{M}_3}$: Cross correlation of X-spectrums of maps M_1 and M_3

$CCY_{\mathcal{M}_1\mathcal{M}_3}$: Cross correlation of Y-spectrums of maps M_1 and M_3

ω : Acceptance index

$\omega(M_1, M_2)$: Acceptance index of maps M_1 and M_2

$arg(M_1, M_2)$: Agreement between maps M_1 and M_2

$dis(M_1, M_2)$: Disagreement between maps M_1 and M_2

List of Abbreviations

GPS	:	Global positioning system
DGPS	:	Differential Global positioning system
CLSF	:	Constrained Local Sub-map Filter
EIF	:	Extended Information filter
EKF	:	Extended Kalman filter
SEIF	:	Sparse Extended Information filter
SLAM	:	Simultaneous localization and mapping
ICP	:	Iterative closest point
CML	:	Concurrent mapping and localization

Chapter 1

Introduction

1.1 Introduction

Building a map is a key requirement for a robot to operate in an unknown environment where prior information of the environment is unavailable. It is necessary for robot to explore the environment and acquire information of the environment from series of sensors in order to build its own map. Map building process requires an accurate estimation of robot pose either with reference to initial position or with reference to partially constructed map. Accurate estimation of robot pose is easy, if it is possible to incorporate GPS or DGPS sensors to robot. But GPS is unavailable for indoor, underground and underwater robotic applications. It is possible to use dead reckoning which integrates odometry data with steering angle data obtained from encoders of the robot wheels to estimate robot poses along the path. Accurate localization based on odometry is not possible due to errors associated with sensor data. These errors can accumulate due to finite sensor resolution, errors introduced by slipping and skidding, kinematical errors due to wear of wheels, wheel misalignment and travelling in uneven surfaces. Some robots use skid steering, thus their odometry data will

always be erroneous.

Features in the environment can be incorporated to eliminates errors from odometry based localization. Therefore map building and self localization problem have to be solved simultaneously. This problem is commonly known as Simultaneous Localization and Mapping(SLAM), also termed as Concurrent Mapping and Localization(CML). Although there are solutions to single robot SLAM problem, employing a single robot to build a map of a large environment is not cost effective and time saving. There may be situations that a single robot will fail to build a map autonomously due to limited battery power or fuel. Therefore deploying multiple robots to build a map of large environment is effective and time saving. Various solutions have been proposed to solve the single robot simultaneous localization and mapping problem, but reasonable research contribution is required to efficiently solve the multi-robot SLAM problem.

1.2 The SLAM Challenges

SLAM challenge is to acquire a spatial model of the robot environment while localizing relative to the acquired model. In SLAM there is bi-directional dependency between map accuracy and pose uncertainty. Sensor limitations, measurement noise, data dimensionality, data association, dynamic environments, obstacles, and explore unknown area are the factors affecting the solution to SLAM problem [1].

To acquire a map of the environment, the robot has to process different sensor data in order to fuse them together. Sensors used in mapping are range finders based on sonar, laser or infrared technology, radar, cameras, GPS, compass and gyroscopes. All these sensors are subjected to range limitations and errors referred to as measurement noise. To overcome the problem of range limitation, robot has to navigate

around its environment to complete mapping. Motion commands are issued to robot in order to navigate around its environment for mapping. Location information of the robot is important since all the sensor readings are referred to those locations and can not be interpreted without location data. Robot motions are subjected to errors due to slipping, uneven terrain, different wheel sizes and errors of encoder readings. Therefore control signals are insufficient to determine robot pose relative to its environment. Because errors in robot control accumulate over time, they affect the way future sensor measurements are interpreted.

Another challenge that is posed to SLAM algorithm is dealing with data dimensionality. Consider how much data it would take to describe a particular environment up to a certain amount of detail. Whenever precision is desired, for example two-dimensional floor plan with good resolution or three-dimensional model of an underwater mapping, more and more details about the environment acquired to be stored. Every additional piece of data increases the memory requirements and time complexity of the mapping and localization algorithm.

As a robot navigate through the environment it potentially visits the same physical location multiple times. In other words different observations are taken at different time intervals that are correspond to the same feature in real world. Therefore there should be a mechanism to identify whether observations correspond to the same feature or not. This identification is called data association (or correspondence problem) which is the hardest problem in robotic mapping. Data association incurs a high risk. Resulting map will have irrecoverable errors if two observations are falsely considered to match to the same feature or vice versa. As robot explores large scale environments requiring longer travel distances, as robot position error grows, the number of possible matches will grow progressively and risk of false data association will be higher.

In many real life circumstances robots can not assume that past or current observations are representing the current state of the world. Consider furniture in a building, position of a door or parked car that were observed in the past but have moved in the mean time. Consider any structural changes in the environment such as adding new building, removing existing building or changes in an existing building and appearance of a tree or environment in different seasons. All of these circumstances are added to the dynamics of the environment and there exists no general solution to deal with dynamics of the environment adequately. Moving obstacles such as people, vehicles will complicate solution to SLAM problem. Robots explore an environment for a constrained period of time which the world is assumed to be static during that time.

Unexplored area of the environment has to be explored to map the area or to solve SLAM problem. During the exploration of the environment, robot must choose their way to explore new area to be acquired. There can be unexpected kind of obstacles like pits to fall into, curbs and other kinds of environmental aspects not accounted. It is typically unknown what challenges will be provided by the environment, therefore robot equipped with sensors and robust behaviors that enable it to deal with unanticipated situations.

1.3 Problem statement

Accurate localization and map building is essential when a robot is introduced to a new environment where no *priori* available information. As robot navigates, it explores new environment while building a map of environment and localizing with respect to partially constructed map. When robot revisits a previously explored area, it can reflect backward the changes appeared to the map. Robot can increases the map

accuracy by revisiting previously mapped area [2–5]. Various solutions are available to SLAM problem in literature [6–18]. Whatever the underline theory for solving SLAM filter, the final map is either feature based or occupancy grid based. Feature-based maps are used in early solutions to SLAM problem. Feature based map suffer from inability to model free and occupied space of the environment. Accurate occupancy grid maps are easily generated with high precision sensors like laser range finders. Occupancy grid based map can be used for path planning and obstacle avoidance. This property encourages to use occupancy grid map in SLAM filter. The main drawback of occupancy grid map is that it only contains information of horizontal cross section of the environment at the hight where the laser range finder(or ranging device) is mounted hence a 2D map. This problem can be eliminated by 3D occupancy grid. But higher memory requirement and computational complexity will degrade the realtime performance of the SLAM filter with 3D occupancy grid.

Research has been carried to solve multi-robot SLAM problem. Majority of them are based on feature based maps [19–26]. Multi-robot SLAM with occupancy grid maps is less addressed in research community [27, 28]. Accurate map merging and closing loops discovered by more than one robot are the main challenges of multi-robot SLAM. Although various solutions are available for single robot SLAM problem, more research is necessary to find solutions for multi-robot SLAM specially with occupancy grid maps.

1.4 Contributions of the thesis

In this thesis, occupancy grid based multi-robot SLAM algorithm is developed. Robots individually builded maps of the environment they have acquired without communication with its team members. At the end of terrain acquisition and map build-

ing by individual robots, central computer or one robot gathers all the maps and merges them to generate complete map of the environment. Multi-robot SLAM algorithm that does not require prior information about relative transformations of the maps, is developed. The algorithm itself several possible candidate transformation. The algorithm merges maps choosing the best transformation among several possible candidate transformation.

1.5 Thesis Overview

Chapter 2 provides an overview to SLAM problem and the state of the art solutions that has been proposed to solve the problem and discusses the qualities and drawbacks of proposed solutions. In addition to present the various solution, chapter 2 overviews the proposed solutions to multi-robot SLAM problem. Chapter 3 detailed background to single robot SLAM problem which is used for multi-robot SLAM in next chapter. Chapter 4 discusses proposed solution for multi-robot SLAM and map merging technique employed to merge maps from multiple robots. Chapter 5 draws the conclusions and suggests future directions for the extension of this work.

Chapter 2

Literature Review

2.1 Fundamentals and Concepts

The problem of simultaneous localization and mapping(SLAM) is the ability to position a robot in an unknown location of an unknown environment and build a consistent map of the environment using observations taken from the sensors of the robot while localizing in the map that is being built. SLAM is also known as concurrent localization and mapping(CML)

Mapping using robots was in the researched for decades and concept of SLAM was introduced to the research community from the work by R. Smith *et al.* [29, 30]. R. Smith *et al.* [31] showed that, estimation of landmarks that are observed by a mobile robot moving in an unknown environment is necessarily correlated with each other due to the uncertainty of the estimated robot pose.

S. Thrun *et al.* [32] propose a probabilistic approach to solve SLAM problem. Probabilistic approach is important in SLAM because both measurements and robot controls noise can be statistically modelled. Commonly used sensors are subjected to errors, often refer to as measurement noise and range sensors have range limitations

[1]. Robot motions are also subjected to errors due to slip, skid, uneven surfaces and different wheel sizes(due to pressure difference in tires). Therefore robot pose relative to its environment can not be determined only using robot controls. Since these errors are accumulated over the time, and they affect the way future sensor measurements are integrated into the map [1]. Pose of the robot is represented by x-coordinate x_R, y-coordinate y_R and heading θ_R. Since the robot motion is inaccurate, effect of robot control u_k to move robot from $x_r(k-1)$ to $x_r(k)$ where,

$$x_r(k) = [x_R \; y_R \; \theta_R]^T \tag{2.1}$$

is the state vector describing the pose of the robot at time k, can be modelled by conditional probability density,

$$p(x_r(k)|x_r(k-1), u(k)) \tag{2.2}$$

which is called motion model. Here it is assumed that robot operates on a planer surface, therefore Cartesian coordinates in the plane and the heading direction is sufficient to model the robot pose.

$$u(k) = [V(k) \; \omega(k)]^T \tag{2.3}$$

is the control vector, applied at time $k-1$ to drive the robot to a state $x_r(k)$ from state $x_r(k-1)$ at time k. $V(k)$ is the translational velocity vector and $\omega(k)$ is the rotational velocity vector. This state transition is assumed to be Markov process in which next state $x_r(k)$ depends only on the immediate proceeding state $x_r(k-1)$ and the applied control $u(k)$ and it is independent from both the observations and the map [32, 33]. S. Thrun *et al.* [32] assume that landmarks are not necessarily

distinguishable or may be entirely indistinguishable in some cases. They also assume that there are errors in range, bearing or type of landmark. Therefore observation model for robot sensors is proposed as,

$$p(z(k)|x_r(k), m) \tag{2.4}$$

$z_i(k)$ is the observation of i^{th} landmark at time k. $z(k)$ denotes the all observable landmarks at time k.

$$m = \{m_1^T \; m_2^T \; ... \; m_N^T\}^T \tag{2.5}$$

is the set of all N landmarks where,

$$m_i = [x_i \; y_i]^T \tag{2.6}$$

is the vector describing the Cartesian coordinates of i^{th} landmark in the world coordinates and it is time invariant. The observation model determines the likelihood of making observation $z(k)$ when the robot is at location $x_r(k)$, assuming that m is the correct model of the environment [32].

$$\begin{aligned} &p(x(k)|z_{1:k}, u_{1:k}) = \\ &\quad \eta p(z(k)|x(k)) \int p(x(k)|x(k-1), u(k)) p(x(k-1)|z_{1:k-1}, u_{1:k-1}) dx(k-1) \end{aligned} \tag{2.7}$$

Where,

$$z_{1:k} = \{z(1), z(2), ..., z(k)\} \tag{2.8}$$

is the history of sensor data measurements(set of all landmark observations),

$$u_{1:k} = \{u(1), u(2), ..., u(k)\} \tag{2.9}$$

is the history of control inputs, $x(k)$ is the set of parameters need to be estimated and η is a normalizer [1].

In SLAM both map and robot pose should be estimated together because errors in robot pose or errors in map affect the integration of future observations to the map. SLAM problem is categorized to full SLAM problem and online SLAM problem. Full SLAM problem is defined as the estimation of the map and complete pose history of the robot using history of sensor data measurements and history of robot controls. The full SLAM problem can be probabilistically defined as $p(x_r(1:k), m|z_{1:k}, u_{1:k})$ where $x_r(1:k)$ is the complete pose history of the robot. Map of the environment and current pose of the robot will be calculated using history of sensor data measurements and history of robot controls.

Online SLAM problem can be defined as the concurrent estimation of the current robot pose and the map of the environment using all past control inputs and measurements. The online SLAM problem can be probabilistically defined as $p(x_r(k), m|z_{1:k}, u_{1:k})$. In contrast, the full SLAM problem is defined as the estimation of the map along with the complete pose history. It is more difficult to estimate full SLAM problem than online SLAM problem due to its high dimensionality of the parameter space and the data association problem, with a large number of pose-feature associations. On account of above reasons online SLAM problem has been widely explored compared to full SLAM problem. This thesis also focuses on online SLAM problem. Therefore $x(k)$ will be,

$$x(k) = [x_r(k)^T \; m^T]^T \tag{2.10}$$

where $x_r(k)$ is robot pose and m^T is map of the environment. Hence Eq 2.7 can be written as,

$$p(x_r(k), m|z_{1:k}, u_{1:k}) = \eta p(z(k)|x_r(k), m) \int p(x_r(k), m|x_r(k-1), m, u(k)) p(x_r(k-1), m|z_{1:k-1}, u_{1:k-1}) dx_r(k-1) dm \tag{2.11}$$

It can be assumed that the world is static, therefore any effect to the parameters referring to variation of map can be neglected. Hence Eq 2.11 will be reduced to,

$$p(x_r(k),m|z_{1:k}, u_{1:k}) = \eta p(z(k)|x_r(k), m) \int p(x_r(k)|x_r(k-1), u(k)) p(x_r(k-1), m|z_{1:k-1}, u_{1:k-1}) dx_r(k-1) \tag{2.12}$$

Where $p(z(k)|x_r(k), m)$ is the observation model, $p(x_r(k)|x_r(k-1), u(k))$ is the motion model and $p(x_r(k-1), m|z_{1:k-1}, u_{1:k-1})$ is the estimation of Eq 2.12 for previous step. This shows Eq 2.12 can be solved recursively.

In SLAM problem there are several methods to represent map. Landmark-based maps and occupancy grid [34] based maps are common methods. Identified features in environment are used as landmarks in landmark based maps. All the necessary parameters should be calculated from raw sensor data. Occupancy grid segments the environment to finite number of regular shaped(square shape in most cases) cells. Each cell contains the probability that particular cell is occupied and sensor data are used to calculate the occupancy probability of the cells. Landmark based mapping method is widely used in SLAM [35].

Work by Dissanayake *et al* [7], proved and verified that SLAM problem can be solved recursively. They formulate a solution using Extended Kalman filter(EKF) to SLAM problem with nonlinear motion models and nonlinear asynchronous observation models.

2.1.1 System Model

System model is used to transform robot from $x_r(k-1)$ to $x_r(k)$ according to control vector. Therefore final pose of the robot is affected by three factors, pose of the robot prior to the application of control, control vector at time k and motion disturbance.

$$x_r(k) = f_r(x_r(k-1) + u(k)) + v_v(k) \tag{2.13}$$

$f_r(.)$ describes the kinematics of the robot and $v(k)$ is zero mean uncorrelated, additive motion disturbance with covariance $Q_v(k)$. The motion disturbance is assumed to be Gaussian in EKF solution to SLAM problem. Eq 2.13 can be expanded as,

$$x_r(k) = x_r(k-1) + \Delta t V(k) \cos\left(\theta_r(k) + \frac{\Delta t \omega(k)}{2}\right) \tag{2.14}$$

$$y_r(k) = y_r(k-1) + \Delta t V(k) \sin\left(\theta_r(k) + \frac{\Delta t \omega(k)}{2}\right) \tag{2.15}$$

$$\theta_r(k) = \theta_r(k-1) + \Delta t \omega(k) \tag{2.16}$$

$x_R(k)$ and $y_R(k)$ are coordinates of the robot potion at time k while $\theta_R(k)$ is the heading of the robot.

Motion model in Eq 2.2 and system model in Eq 2.13 are interrelated [33].

$$P(x_r(k)|x_r(k-1), u(k)) \Longleftrightarrow x_r(k) = f(x_r(k-1) + u(k)) + v(k) \tag{2.17}$$

2.1.2 Measurement Model

Range and bearing to a landmark is the commonly used sensor model in SLAM. There are examples of bearing only SLAM which mostly use computer vision techniques. Observation of i^{th} landmark in range and bearing sensor model can be expressed as,

$$z_i(k) = \begin{bmatrix} r_i(k) \\ \theta_i(k) \end{bmatrix} = h_i(x(k)) + w(k) \tag{2.18}$$

$r_i(k)$ is the range to i^{th} landmark and $\theta_i(k)$ is the bearing to i^{th} landmark. $h(.)$ models the geometry of the observation and $w(k)$ is the zero mean uncorrelated measurement error with covariance $R(k)$. The measurement error is assumed to be Gaussian in EKF solution to SLAM problem. $r_i(k)$ and $\theta_i(k)$ can be calculated using,

$$r_i(k) = \sqrt{(x_i - x_r)^2 + (y_i - y_r)^2} + w_r(k) \tag{2.19}$$

$$\theta_i(k) = \arctan(y_i - y_r, x_i - x_r) - \theta_r + w_\theta(k) \tag{2.20}$$

where $w_r(k)$ and $w_\theta(k)$ are zero mean uncorrelated measurement error related with range measurement and bearing measurement respectively.

Observation model in Eq 2.4 and measurement model in Eq 2.18 are interrelated [33].

$$P(z(k)|x_r(k), m) \Longleftrightarrow z_i(k) = h_i(x(k)) + w(k) \tag{2.21}$$

2.2 FastSLAM

FastSLAM is an algorithm that integrates both particle filter and EKF to solve SLAM problem [36]. Landmark estimation and robot pose estimation depend on each other due to the uncertainty of the robot pose. If it is possible to obtain the true path of the robot by a reliable mechanism, then the problem of landmark position estimation will be independent of each other. In other words, if the true path of the robot is known, location information of one landmark does not carry any information about other landmarks [37]. Simply landmarks are uncorrelated given that true path of the

robot is available. This factorization of path estimation and landmarks estimation was introduced to the robotics community by Murphy [38]. According to Murphy's work, SLAM posterior can be rewritten as,

$$p(x_r(1:k), m|z_{1:k}, u_{1:k}) = p(x_r(1:k)|z_{1:k}, u_{1:k}) \prod_{n=1}^{N} p(m_i|x_r(1:k), z_{1:k}, u_{1:k}) \quad (2.22)$$

where N is the number of landmarks. Due to above factorization SLAM problem can be decomposed to one estimator over the robot path and N independent landmark estimators. All together there are $N+1$ recursive estimators, one particle filter for robot path and N EKFs for landmarks. If M particles are employed to solve SLAM problem, there will be $M \times N$ Extended Kalman filters. Each particle in the filter represents the current robot pose, mean position of the landmarks and their variances.

When robot moves to a new pose, filter proposes a new robot pose to all the particles. Since samples can not be directly drawn from the SLAM posterior at time t, samples are drawn from simpler distribution called proposal distribution. The proposal distribution of FastSLAM generates guesses of the robot pose at time t, using previous robot pose. This guess is obtained by sampling the probabilistic motion model Eq 2.2 which can be any non-linear function and needs not to be linearized to draw samples.

The next step of FastSLAM is landmark location estimation of each and every particle in the particle filter. FastSLAM represents the conditional landmark estimates $p(m|x_r(1:k), z_{1:k}, u_{1:k})$ by extended Kalman filters. This estimate is conditioned on the robot pose, therefore extended Kalman filters are attached to particles in particle set. Linearized version of the observation model (in Eq 2.4) is used in landmark update equations since they are based on EKF which approximates the observation model using linear Gaussian function. The resulting distribution

$p(m|x_r(1:k), z_{1:k}, u_{1:k})$ is Gaussian due to linear Gaussian observation model and independent of the nonlinear motion model. This is a consequence of the use of sampling to approximate the distribution over the robot's pose.

Samples drawn from proposal distribution are distributed according to $p(x_r(1:k)|z_{1:k-1}, u_{1:k})$ and they do not match the desired posterior $p(x_r(1:k)|z_{1:k}, u_{1:k})$. The difference is corrected during the importance sampling. Before importance sampling, each particle is given an importance weight(w) and it is a measure of how a particle agrees with the observations.

$$w_i = \frac{\text{target distribution}}{\text{proposal distribution}} \tag{2.23}$$

The last step of FastSLAM is importance re-sampling. New set of samples are drawn from the present set with replacements and replacement probabilities are proportional to importance weights. In other words, re-sampling eliminates samples with low importance weights and multiplies samples with high important weights. Finally particles which comply with observations will survive.

Each and every instance the robot progresses to new pose, above procedure is repeatedly applied to the particles. This procedure converges asymptotically to the true posterior as the number of particles(M) goes to infinity but lesser number of particles are sufficient for a practical implementation of FastSLAM [37].

The key advantage of FastSLAM algorithm is the ability to incorporate multi-modal beliefs and nonlinear motion and observation models increase the robustness of the particle filter based solution. Data association is calculated on per particle basis, this enables filter to maintain several hypotheses of data association. Therefore FastSLAM is much more robust to the errors occurred due to wrong data associations. Time complexity of FastSLAM algorithm is $\mathcal{O}N(log(N))$ therefore it can handle maps

with large number of landmarks [8].

There are two versions for FastSLAM namely FastSLAM 1.0 and FastSLAM 2.0. The main differences between these two algorithms are choice of proposal distribution and calculation of importance weights. The landmark updates, data association and re-sampling are remain unchanged.

FastSLAM 1.0

Samples for robot pose can not be drawn directly from the SLAM posterior at time t. Therefore samples are drawn from simpler distribution called proposal distribution and the difference will be corrected at re-sampling step. The proposal distribution of FastSLAM 1.0 generates guesses of the robot's pose at time t for each particle. This guess is obtained by sampling from the probabilistic motion model which can be any non-linear function [8].

Calculation of importance weight is performed according to Eq. 2.23 and it is a measure of how well the proposal distribution approximates the target distribution. After simplification of Eq. 2.23 using Bayes rule the observation likelihood is the weight in FastSLAM 1.0 [8].

FastSLAM 2.0

FastSLAM 1.0 samples the robot pose according to the motion model and it does not incorporate the observation for proposal distribution. Observation is introduced in calculation of importance weights for re-sampling. This approach is suboptimal if the noise in the motion is large relative to the measurement noise. This will be the typical scenario when the robot is equipped with a laser range finder. In such situations, sampled poses will mostly falls into areas of low measurement likelihood and will subsequently be terminated in the re-sampling phase with high probability. Therefore

observation z_t at time t is incorporated to calculate the proposal distribution [10].

Calculation of importance weight is complicated in FastSLAM 2.0 and it depends on the measurement model, landmark estimate at time $t-1$ and the motion model.

2.2.1 Grid Based FastSLAM

FastSLAM has been extended beyond conventional landmark list. Grid based FastSLAM replaces the landmark list by an occupancy grid. Grid based FastSLAM does not require pre defined definition of landmarks and any arbitrary environment can be modelled using occupancy grid. Each particle has a full occupancy grid of the environment instead of landmarks.

D. Hahnel *et al* [11] propose a new algorithm that combines Rao-Blackwellized particle filtering and scan matching. Scan matching is used to transform sequence of laser scans into odometry measurements that minimizes the odometric errors during mapping. In this way number of particles can be reduced so that map of even larger environment can be constructed online. The corrected odometry and laser scans are used for map estimation in the particle filter. A probabilistic model of the residual errors of scan matching process is then used for re-sampling steps. The lower variance in the corrected odometry reduces the number of necessary re-sampling steps and decreases the particle depletion problem that typically prevents the robot from closing larger loops.

C. Stachniss *et al* [2] and [16] propose a novel technique that combines autonomous exploration and simultaneous localization and mapping. For SLAM algorithm authors used the grid based version of FastSLAM algorithm described in [11] and Frontier-Based exploration is used to explore new terrain. Each particle has an occupancy grid map and a topological map and they are updated when the robot is

performing exploration. In the topological map, the vertices represent the locations visited by the robot and the edges represent the trajectory corresponding to the particle. After adding new node to the topological map, algorithm performs a ray-casting operation in the occupancy in order to check whether the new node is visible from another node. If it detects another node and if the length of the shortest path from the current pose to the previously visited node in topological map was large, then these shortcuts represent edges that would close a loop in the topological map. Algorithm uses these shortcuts for loop closing whenever the uncertainty of the robot pose becomes too large. If uncertainty is too large then the algorithm starts loop closing behavior and re-visits the portions of previously explored area in order to reduce the uncertainty while closing the loop. Since it is time consuming task to perform the above mechanism and find the possibility of closing loop for all particles in on line processing, the check has been limited to the particle with highest importance weight. During loop closing uncertainty in the pose reduces because robot navigates through the previously explored area localizing itself in the map constructed so far. Therefore it is unlikely that particles vanish during loop closing. But too small uncertainty in robot pose become problematic when there is nested loop in the map. If robot eliminates uncertainty while closing inner loop, particle necessary to close outer loop may vanish and filter will diverge and robot fails to build a correct map. To avoid this problem, algorithm introduces a threshold for uncertainty of the robot pose to stop loop closing. Whenever uncertainty becomes smaller than given threshold robot stops loop closing and resumes terrain acquisition. This algorithm can build map on line actively closing loops. Although there is a threshold value for stoping loop closing, it does not grantee that filter will not diverge. Whole algorithm has to be repeated in case of filter divergence. Comparison with the method in [11], each particle in this algorithm maintains a topological map of the environment. These topological maps

are stored in a graph data structure therefore memory requirement is low as compared to the memory requirement where grid map(Occupancy grid) is employed. But ray-casting operation in adding new node and applying A^* algorithm or Dijkstra's algorithm for computing shortest path add significant computational burden.

G. Grisetti *et al* [17] proposed an algorithm for Rao-Blackwellized particle filter based grid SLAM with improved proposal distribution and adaptive re-sampling. FastSLAM 2.0 uses Rao-Blackwellized particle filter that uses Gaussian approximation of the improved proposal for feature(Landmark) based SLAM. It is assumed that the error affecting the feature detection is Gaussian. [17] extends the idea of computing an improved proposal distribution for grid based SLAM. The proposal distribution is designed to generate new pose that agrees with both current control and current observation. Algorithm uses scan matching to determine the mode of the meaningful area of the observation likelihood and sample around the mode to calculate the proposal distribution. There is a possibility to fail scan matching due to poor observation or due to small overlapping area between current laser scan and previously computed map. In such cases the raw motion model of the robot is used but there is rare opportunity to occur such situations. Re-sampling has a major influence on the performance of the particle filter. In re-sampling particles with low importance weights are replaced by those with high importance weight but there is a risk of replacing better matched particles that leads to particle depletion. On the other hand re-sampling is important since target distribution is approximated by finite number of samples. Technique described in [39] was employed to calculate effective sample size(ESS) to reduce the risk of particle depletion. Algorithm in [39] calculates the ESS and re-sampling step is postponed until ESS falls below the given threshold. With above improvements, results show that the number of particle required for a given data set is drastically reduced compared to the algorithm discussed in [11].

S. Grzonka *et al* [18] proposed a new algorithm that uses look-ahead proposals for grid based, Rao-Blackwellized particle filter SLAM. Rao-Blackwellized particle filter SLAM will fail when the particle distribution is significantly different from the true posterior. This can happen due to either proposal distribution provides a bad approximation of the true posterior or the environment does not provide enough structure to allow proper particle weighting. Such failures can be avoided by either using a large number of particles or directing the fewer number of particles to more accurate position. Authors in [18] propose an algorithm to direct the fewer number of particles to more accurate position. Pose prediction is computed based on the next k sensory inputs instead of just one immediate input. In other word, k measurements are used to better localize each particle within its own map. In the algorithm there are two types of particles namely SLAM particles and localizing particles. SLAM particles are the conventional particles in Rao-Blackwellized particle filter SLAM while localizing particles are used for look-ahead proposals. At the time of calculation of new proposals, each SLAM particle is assigned number of localization particles(N) based on motion model. If the filter has M, SLAM particles, altogether there will be $M \times N$ localization particles in the filter. Each localization particle is moved forward to collect k sensory inputs in the map of corresponding SLAM particle using next k control signals and observe the likelihood how future measurement are agreed with current map. Then likelihoods of observations are propagated along the trajectory backward in time. All the localization particles are weighted according to the propagated likelihood of observations. Highest M localization particles are selected in re-sampling step and they will be the SLAM particles for next step of the filter. There are $M \times N$ particles and only M particles will survive at the end. Therefore, there is a possibility that none of the localization particles own to some SLAM particles will not be selected while multiple localization particle own to one SLAM particle to be

selected. There is an optimum number for k look-ahead steps and further growing k will not improve the accuracy. This is due to fact that localization particles will move to unknown area of the map at time t and it is unable to calculate the likelihood of observation.

2.3 Other Solutions to SLAM Problem

2.3.1 DP-SLAM

In [12] and [14] the authors propose a novel approach based on Rao-Blackwellized particle filter to SLAM problem namely Distributed Particle Simultaneous Localization and Mapping(DP-SLAM). DP-SLAM is based on occupancy grid map, therefore it does not require feature extraction and data association. There is only a single map in DP-SLAM and all the particles are associated with it. Each and every grid square of the occupancy grid holds a tree structure which maintains the IDs of the particles that have made changes to the occupancy of the relevant grid square. The grid is initialized with empty tree and associated tree is updated whenever a particle makes an observation about a grid square. This method allows particles to behave as if they have their own maps. DP-SLAM maintains an ancestry tree of all of the particles and it contains all of the current particles as leaves. Each particle maintains a list of grid squares that it has updated. The parent of a given node of ancestry tree represents the particle of the previous iteration from which that particle was resampled. Algorithm maintains a bounded size ancestry tree by removing unnecessary branches of the tree.

DP-SLAM it self can close loop without explicit use of loop closing algorithms or heuristics but algorithm needs large number of particles to close loop. Data retrieving

from a grid cell is far more complicated compared to conventional occupancy grid on account of memory utilization technique.

2.3.2 EKF-SLAM

Extended Kalman filter(EKF) is widely used in solving SLAM problem. Kalman filter is used to represent posteriors $p(x_r(k), m|z_{1:k}, u_{1:k})$ with Gaussian. It is assumed that noise in robot motion and perception(observation) is Gaussian and the initial uncertainty of robot pose must be Gaussian in Extended Kalman filter solution to SLAM problem.

$$x(k) = f(x(k-1) + u(k)) + v(k) \tag{2.24}$$

State Vector and Covariance Matrix

In EKF terminology the set of parameters need to be estimated is called state vector. In EKF-SLAM $x(k)$ in Eq 2.10 is the state vector. The elements of Eq 2.10 are called state variables. All the sensor observations are integrated into a single covariance matrix in EKF solution to SLAM problem. The covariance matrix of the state vector $\Sigma(k)$. Covariance matrix contains three components. They are covariance of robot pose $\Sigma_{rr}(k)$, covariance of the map $\Sigma_{mm}(k)$ and cross covariance between robot pose and map $\Sigma_{rm}(k)$. Covariance matrix will be,

$$\Sigma(k) = \begin{bmatrix} \Sigma_{rr}(k) & \Sigma_{rm}(k) \\ \Sigma_{rm}^T(k) & \Sigma_{mm}(k) \end{bmatrix} \tag{2.25}$$

Estimation Process

In general, estimation using Kalman filter proceeds recursively. Three are steps namely prediction, observation, and update [7]. In Kalman filter literature any quantity with accent hat is an estimated(or predicted) quantity. For example $x(k)$ and $z(k)$ are the state vector and observation, while $\hat{x}(k)$ and $\hat{z}(k)$ are the estimate for the state vector and predicted observation respectively. Any quantity with superscript "$-$" is predicted or priori value while the same quantity with superscript "$+$" is updated or posteriori value. For example $\hat{x}^-(k)$ and $\hat{x}^+(k)$ are predicted and updated estimates of the state vector respectively. $\Sigma^-(k)$ and $\Sigma^+(k)$ are priori and posteriori covariance matrix respectively.

Prediction

In the prediction stage, filter generates predictions for state estimate, observation and covariance matrix using updated state estimate and posteriori covariance matrix of previous state. $\hat{x}^-(k)$ will be calculated using 2.24.

$$\hat{x}^-(k) = f(x^+(k-1) + u(k)) \tag{2.26}$$

$\hat{z}(k)$ can be calculated using 2.18 and $\hat{x}^-(k)$

$$\hat{z}(k) = h(x^-(k)) \tag{2.27}$$

The covariance matrix must also propagate uncertainty of state estimates and robot controls in prediction state. The EKF linearizes the propagation of uncertainty about the current estimate $\hat{x}^+(k-1)$ using the Jacobian $\nabla_x f(k)$ of f evaluated at $\hat{x}^+(k-1)$ and uncertainty of control input $u(k)$ using the Jacobian $\nabla_u f(k)$ evaluated around

$u(k)$. Therefore priori covariance matrix, $\Sigma^-(k)$ can be written as

$$\Sigma^-(k) = \nabla_x f(k)\Sigma^+(k-1)\nabla_x f^T(k) \; + \; \nabla_u f(k)\Sigma_u \nabla_u f^T(k) \; + \; Q(k) \tag{2.28}$$

where,

$$\Sigma_u = \begin{bmatrix} \sigma_V^2 & 0 \\ 0 & \sigma_\omega^2 \end{bmatrix} \tag{2.29}$$

The Jacobians $\nabla_x f(k)$ and $\nabla_u f(k)$ can be represented as follows [35].

$$\nabla_x f(k) \;\; = \;\; \begin{bmatrix} \frac{\partial f_r}{\partial\{x_r, y_r, \theta_r\}} & 0_{rm} \\ 0_{mr} & I_{mm} \end{bmatrix} \tag{2.30}$$

$$\nabla_u f(k) \;\; = \;\; \begin{bmatrix} \frac{\partial f_u}{\partial\{V,\omega\}} \\ 0_{m\times 2} \end{bmatrix} \tag{2.31}$$

Therefore, Σ^- can be efficiently implemented by,

$$\Sigma^-(k) = \begin{bmatrix} F_x \Sigma_{rr}^-(k-1) F_x^T + F_u \Sigma_u F_u^T & F_x \Sigma_{rm}(k-1) \\ (F_x \Sigma_{rm}(k-1))^T & \Sigma_{mm}(k-1) \end{bmatrix} \tag{2.32}$$

where $F_x = \frac{\partial f_r}{\partial\{x_r, y_r, \theta_r\}}$ and $F_u = \frac{\partial f_u}{\partial\{V,\omega\}}$.

Observation

When new observation is received from the robot sensors, it is associated with observable features in the environment. The difference between the actual observation $z(k)$ and the predicted observation $\hat{z}^-(k)$ is defined as innovation $\nu(k)$.

$$\nu(k) = z(k) - \hat{z}^-(k) \tag{2.33}$$

The innovation covariance, $S(k)$, is computed using priori covariance matrix of state estimates, the Jacobian of the observation model, $\nabla_x h(k)$, and the covariance of the observation, $R(k)$.

$$S(k) = \nabla_x h(k) \Sigma^-(k) \nabla_x h^T(k) \; + \; R(k) \tag{2.34}$$

Update

The state estimate and state covariance matrix are updated in the update step.

$$\hat{x}^+(k) \;\; = \;\; \hat{x}^-(k) \; + \; W(k)\nu(k) \tag{2.35}$$

$$\Sigma^+(k) \;\; = \;\; \Sigma^-(k) \; - \; W(k)S(k)W^T(k) \tag{2.36}$$

where $W(k)$ is gain matrix.

$$W(k) = \Sigma^-(k) \nabla_x h^T(k) S^{-1}(k) \tag{2.37}$$

Limitations of EKF-SLAM

Although EKF-SLAM is a good solution to SLAM problem, it suffers from two major problems. They are quadratic computational complexity and sensitivity to failures in data associations. The uncertainty of SLAM posterior is represented by a covariance matrix. Correlation matrix has correlations of all possible pairs of state variables(Robot pose and landmarks). For a 2D map with N landmarks, the size of correlation matrix will be $(2N + 3) \times (2N + 3)$ where memory requirement is $\mathcal{O}(N^2)$. Since correlation between all pairs of state variables are maintained, it is necessary to update the correlations of all the state variables whenever observation is made. Therefore number of computations grows quadratically with the number of

landmarks. In EKF-SLAM both computational complexity and memory requirement increases quadratically with the number of landmarks in the map [33, 37]. $\mathcal{O}(N^2)$ complexity limits the maximum number of landmarks in the map. Therefore it is necessary to modify the original EKF-SLAM algorithm if it is necessary to use in large environments. In EKF-SLAM, maximum likelihood is used to verify whether an observation belongs to existing landmarks in the state variables or not. New landmark is assigned if there is less probability that an observation belongs to an existing landmarks. There is no way of representing the uncertainty of data association in EKF-SLAM formulation. Therefore effect of wrong data association can not be recovered by any means in future stages of the filter. Therefore large number of errors in data association will diverge the SLAM filter [37]. Map generated by EKF-SLAM contains the mean of estimation of the landmarks and their variances, therefore those map does not model free and occupied space.

2.3.3 Sparse Extended Information Filter Based SLAM

S. Thrun *et al* [13] propose a solution to SLAM problem using an extended version of the information filter (EIF) whose estimation process is similar to the EKF. The information matrix is the central data structure in the EIF which is equivalent to inverse of the covariance matrix of EKF. The elements in the information matrix represent either links between robot pose and landmarks or the landmarks themselves. The normalized information matrix is almost sparse [13]. The information matrix in SEIF is sparse. At the given instant robot can only observe limited number of landmarks known as active landmarks. When the robot makes new observation, elements in related to robot pose and observed landmarks in the information matrix are affected. Filter does not update links with robot pose and unobserved landmarks

or observed landmarks and unobserved landmarks. Number of computations need to update the information filter is depend on the number of observed landmarks at that time. It is independent of total number of landmarks. According to S. Thrun *et al* [13] updates are additive. Maximum likelihood approach described in Dissanayake *et al* [7] is used here for data association. SEIF performs lower accuracy compared to EKF solution but the computational speed is improved. Problems in the EKF SLAM such as, $\mathcal{O}(N^2)$ memory requirement, inability to incorporate multiple data association hypotheses and use of Taylor expansion for linearization persist in SEIF based SLAM [13].

2.4 Multi-Robot SLAM

This section reviews research on multi-robot SLAM. Solutions are available for multi-robot SLAM based on: extended Kalman filter, sparse extended information filter, particle filter, constrain local sub-map filter, and D-SLAM framework. Some algorithm requires known initial poses of every member in the team while other algorithm requires either pre-arranged or random meetings among the members of the team during the process terrain acquisition. There are solutions which do not need any of the above conditions. Above techniques are reviewed considering computational complexity, memory requirement and ability to work in realtime. Summary of above techniques is available at the end of the review.

EKF Based Multi-robot SLAM

E.W. Nettleton *et al* [23] propose decentralized multi vehicle map building algorithm based on information filter for multiple unmanned aerial vehicles(UAV). Known initial UAV poses and unique landmark signatures are assumed and those assumptions

restrict the generalization of the algorithm. GPS, Radar, laser range finder, and camera are used as sensors. Main contributions of the paper are decentralized data fusion, effective communication among team members to update their observations and making the system robust to communication failures or changes in communication media.

Employability of EKF and SEIF for Multi-robot SLAM

E.W. Nettleton *et al* [24] discuss the ability of incorporating state space(Kalman filter based) and information based(Information filter based) solution to multi-robot SLAM. It is found that information filter based solutions have advantages than Kalman filter based solutions to multi-robot SLAM problem. Main advantage is the additive nature of the information filter update. In other words, total map is simply the sum of contributions from each individual map. Main drawback of information filter based solution is that, there is no actual map maintained and inverse of the map is maintained. Inversion of the information filter is always required whenever it is necessary to extract any physical location. It is not possible to delete row or column of information matrix since off diagonal terms affect the inversion. Inversion to the state space, performing necessary calculations and returning to information space are necessary to resolve map management issues.

Constrained Local Sub-map Filter Based Multi-robot SLAM

S.B. Williams *et al* [22] propose Constrained Local Sub-map Filter(CLSF) to multi vehicle SLAM. CLSF crates a local sub-map of the immediate vicinity of the robot. This local map is periodically fused into global map of the environment. This representation reduces the computational complexity of global map estimate because update of local covariance matrix is a function of the number of features in the local

sub-map and not the number of landmarks in global map. Uncertainties of the robot pose and feature estimation of local frame of reference tend to be comparatively small, therefore CLSF improves the data association process. Moreover the decision of data association of global map can be deferred until an local sub-map of the environment is available. Correspondences between two maps can be established by comparing the inter feature distances. Transformation between two maps can be found once correspondences are known. This transformation is used as an initial guess for for least squares solution to the transformation between two maps. Least square solution is used to transfer new features into related frame of reference. Algorithm has been verified with simulation results.

Sparse Extended Information Filter Based Multi-robot SLAM

Works by S. Thrun *et al* [19] use Sparse Extended Information Filter(SEIF) based solution to multi robot SLAM problem. Proposed algorithm enables a team of robots to build joint map without prior knowledge of relative starting positions and ambiguous of landmark positions. There is no periodical fusion of local map with the global map like in [22]. Fusion is done only after completing all the local maps. Additivity and locality are the two key properties of the SEIF based multi robot SLAM. Additivity enables multiple robots to integrate their information by adding increments. Locality ensures that all updates performed by a robot are limited to own pose and landmark history. Fusion of local maps to one global map is challenging because each robot maintains its own local coordinates frames and relative position of each other is unknown and complexity of establishing correspondence among landmarks common to several maps. To find good alignment, it searches for corresponding pairs of local landmark configurations in different maps. For each and every identified landmark, algorithm identifies three adjacent landmarks that fall within a small radius. The rel-

ative distances and angles in these triplets are saved in an SR-tree to facilitate easy retrieval. Landmarks with similar local configurations can be identified by searching SR-tree. Correspondences found in this search served as a starting hypothesis for the map fusion. The possible data associations are searched recursively assuming and un-assuming correspondences between landmarks in the different maps. Map fusion is finalized with reduction of overall map likelihood that comes from equating two landmarks and the increase in likelihood that results from the fact that if there were really two separate landmarks. To perform the latter parts of the map fusion, both robot must have detected the landmarks and sensor model is employed to characterize the not seeing a landmark.

Manifold Representation Based Multi-robot SLAM

A. Howard *et al* [28] propose a solution to SLAM problem using manifold representations. Manifold will develop a spiral structure with same location repeated over and over again. In other words, the same location in the world may be represented more than once in the manifold. Loop closure algorithm is relatively expensive in SLAM and refitting the entire map is nontrivial. Loop closing has to perform more than once if the robot moves more than once in the same loop. In order to close loop, one must find out two point clouds in the map which are representing single point cloud in real world. Manifold representation has lazy loop closure and loop closing can be indefinitely delayed until algorithm is confident enough to close loop without risking map consistency. In addition to delaying the decision of loop closing, loop closing algorithm is executed only once for each loop in the environment. Island merging algorithm is used to fuse two maps obtained from two robots. Island merging is performed whenever two robots identify themselves (authors used the term mutual observation)to have their relative pose. Two islands are considered to be rigid and

quickly bring them into rough alignment and fine tune the merging. Island merging is computationally expensive and needs to repeat $N-1$ times for group of N robots. Robots are necessary to be halted during island merging or loop closing. The algorithm is centralized and has some key limitations. Algorithm is sensitive to communication failures and scales poorly with increasing team size. Authors carried out experiments using only four robots. Developing a distributed algorithm is challenging and it may also be sensitive to communication failures. Algorithm uses scan matching and finally occupancy grid map which can be readily used for path planning and navigation will be available for future use.

EKF and Robot Rendezvous Based Multi-robot SLAM

Works by X.S. Zhou *et al* [21] use robot rendezvous to solve multi robot SLAM problem with unknown initial correspondence. This approach requires to meet robots at least once either as a random or as an arranged meeting by two robots. Extended Kalman Filter(EKF) is used to estimate robot and landmarks' positions and final map is a feature based map. Map alignment problem can be solved by processing relative distance and bearing measurements while two robots are in the sensing range of each other. Transformation between two local maps can be found using robot to robot measurements. There may be possibility of having duplicate landmarks in the merged map due to uncertainty of the landmark estimation. To increase the quality of the merged map authors use *Sequential Nearest Neighbor Test* to detect and combine duplicate landmarks. If landmark L_i and l_i in map m_1 and m_2 represent a same landmark, their distance in the merged map should be either zero in ideal merging or close by in duplicate landmark case. If the algorithm finds any doubt of duplicate landmarks, Mahalanobis distance between two landmarks is calculated. If it is smaller than a threshold then they are considered as duplicate landmarks. Kalman filter is

updated by eliminating duplicate landmarks. This can be easily done by deleting row of the state vector and row and column of the covariance matrix. This works well if the landmark error is smaller than the distance between two landmarks. Decision of matching two landmarks is irrecoverable. To avoid false removing landmarks, detection of duplicate landmarks are started in the vicinity of two observed robots. Most certainly errors of landmark positions are small in the vicinity of two robots and correct matches are most likely to be found there. As new matches are completed and state vectors are updated sequentially, the errors in the landmarks further away are also reduced. Thereafter, it is more likely to find correct matches in distance areas and end up with a consistent map.

D-SLAM Framework Based Multi-robot SLAM

Z. Wang *et al* [20] introduce a solution to multi-robot SLAM problem using D-SLAM frame work. Each robot in the team acquires a map of the environment using Extended Kalman Filter (EKF) SLAM algorithm and D-SLAM framework is used to fuse maps obtained from multiple robots. Corresponding features of two maps which are going to merge are searched in order to find a map alignment hypothesis. Energy function is used to find out corresponding points and correspondence matrix is updated. Energy function is capable of removing outliers and it has components to prevent removing too many points as outliers. Moreover there is possibility to apply constrain on transformation of points one map to the other map. Coarse alignment of two maps can be found based on corresponding features. Algorithm performs joint compatibility test to verify feature correspondence. Feature pairs with large values in correspondence matrix are confident correspondences but those with small values unconfident correspondences. Unconfident correspondences can occur due to outliers, close by features or error accumulation in robot motion. Unconfident correspondences

indicate possible matching and algorithm finds possible features of the global map which are close enough to current unconfident correspondences. Only simulation results are presented in [20]. Algorithm works well for simulation. Data recorded from single robot has been divided into three parts and used as they were from three robots. Results are compared with multi-robot SLAM in single robot EKF-SLAM covering whole area using same data set. Simulation results have same accuracy for single robot and multi-robot SLAM. Although robots individually build maps, integration of local maps to one global map is a centralized.

Particle Filter Based Multi-robot SLAM

A. Howard [27,40] used particle filter to solve multi-robot SLAM problem which can be considered as an extension to the work done in S. Thrun [41]. Either known initial poses of all the robots or robot rendezvous are pre requirement to determine their relative poses among robots. It is assumed that robots are capable of mutual recognition and relative pose determination for nearby robots within line of sight and broadcast actions and observation pairs over a reliable wireless link. Robot does not know presence and poses of the other robots at the time of start, therefore it performs single robot SLAM until it observes another robot. Whenever two robots encounter each other($(i-1)^{th}$ robot observes i^{th} robot) , they determine their relative pose. Then particle filter of the $(i-1)^{th}$ robot is augmented with two additional variables $\langle x^i, \bar{x}^i \rangle$ representing causal and acausal instances of the i^{th} robot and vice versa. Then the queued data for robot i is divided to 2 queues namely causal queue and acausal queue. The causal queue contains the data(odometry and observation) recorded after the time of encounter while the acausal queue contains the data recorded before the time of encounter. The causal queue is empty at the beginning but it is updated with the odometry and observation data of the i^{th} robot received through wireless link.

The causal queue is used to incorporate the observations of the i^{th} robot to the map of $(i-1)^{th}$ robot after the time of encounter. The acausal queue is used to incorporate the observations of the i^{th} robot to the map of $(i-1)^{th}$ robot before the time of encounter. Integration of data in the acausal queue is considered to be a virtual robot moving backward in time until queue becomes empty. If there were any robot which already has the state variable of another robot at the time of encounter a new robot all the details of both robot will be transferred to the new robot. Particle filter of the new robot will be augmented by the variables corresponding to both robots. Above procedure is repeated whenever two robots meet each other. In this manner one robot can incorporate the observation of all the robots into a single map. Finally each and every robot will produce comparable maps of the whole area. No map merging is necessary since all the robots maintaining a map of whole environment. Increasing number of robots will limit the ability of online processing and system is sensitive to communication failures.

2.4.1 Summary of Multi-Robot SLAM

	EKF-MR-SLAM	CLSF	SEIF	Manifold Representation	EKF-Rendezvous	DSLAM	Particle Filter
Complexity	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	$\mathcal{O}(Nlog(N))$	Depend on map resolution	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	Depend on map resolution
Data Association	Necessary	Necessary	Necessary	Not Necessary	Necessary	Necessary	Not Necessary
Memory Requirement	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	Depend on map resolution	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	Depend on map resolution
Communication During Mapping	Necessary	Necessary	No	Necessary	Necessary	No	Necessary
Realtime Operation	In Simmulations	No Information	No Information	Yes Limited robots	Yes Max 2 robots	No Information	Yes Max 4 robots
Known Initial Pose	Yes	Yes or Rendezvous	No	Yes or Rendezvous	No but Rendezvous	No	Yes or Rendezvous
Map Representation	Feature-based	Feature-based	Feature-based	Occupancy grid	Feature-based	Feature-based	Occupancy grid
MR Loop closing	Not Discussed	No	Not Discussed	Yes	Not Discussed	Not Discussed	No
Map Merging	Not Discussed	Yes	Yes	Yes	Yes	Yes	No

Table 2.1: Classification of multi-robot SLAM solutions

Chapter 3

Grid Based Fast SLAM

3.1 Map Representation

3.1.1 Feature Based Maps

Feature based maps use geometric primitives such as corners, points, lines to represent the environment. After each and every measurement cycle, map is updated with these geometric primitives. Introduction of new geometric primitives will increase the complexity of the map. Another representation of the environment, which uses lesser number of geometric primitives, called features has been introduced in order to avoid the complexity. Although feature based maps are efficient way of representing the environment, they are limited to parametric landmarks or modelled objects. Feature based maps suffer from inability to represent complex environments such as natural structures or space between the features. There is no way to represent free space in feature based maps [42].

Features are not readily available after the observation of the environment. Line detection, edge detection or some other technique should be employed to extract

features. Therefore, feature based maps suffer from computational complexity of the feature detection. Realtime performance of online mapping will be affected by the complexity of feature extraction algorithm. Line detection for feature extraction will perform well in man made environments such as indoor applications or out door applications rich of straight line features, but the same algorithm will fail if there is no straight line features in the environment. In general, algorithm does not know which features should be detected in advance. After the feature is detected and location is estimated, the feature based map itself does not maintain any uncertainty of that feature. Due to complexity of the feature and quality of the sensor, there is a risk of resulting wrong or missing features [42, 43].

3.1.2 Topological Maps

Topological maps model the environment without implicit use of metric information. Topological maps are generally represented by graphs whose nodes are distinctive locations and arc that connects nodes denote path information between locations. Whenever topological maps are used, localization and navigation have to be performed using location recognition algorithms. Localization using topological maps performs well in structured environments where distinctive locations are more frequent. In unstructured environments where location recognition is more complex, there is more risk of fail to localize [42, 43].

3.1.3 Occupancy Grid Maps

Two dimensional occupancy grid maps which are also known as evidence grid maps were introduced by H. P. Moravec *et al* [44]. In this representation, the environment is subdivided into a grid of rectangular or square cells. Resolution of the environ-

ment representation directly depends on the size of the cells. Each and every cell is associated with probabilistic measure of occupancy, additional to this grid of the environment. Each cell contains any real number in the interval [0, 1] and that number describes the possible cell states, namely occupied, unoccupied and unknown. Occupancy probability of 1 means it is definitely occupied while occupancy probability of 0 means it is definitely unoccupied. Furthermore occupancy probability of 0.5 declares unknown state of occupancy. Occupancy grid is an efficient approach for fusing multiple sensor measurements and representing uncertainty. It also allows to incorporate different sensor uncertainties. Map resolution depends on the size of the grid cell. Smaller grid cells increase the map quality but it increases memory requirement and number of computations. Occupancy grids represent both occupied space and unoccupied space of the environment, which are useful for path planning and obstacle avoidance [34, 42 44].

3.1.4 Comparison of Map Representations

The aim of the SLAM is to obtain a map of the environment to use in future applications. If the map represents occupied and unoccupied areas, path planning and obstacle avoidance can be done easily. Feature based maps and occupancy grid based maps provide metric information about the environment while topological maps do not provide metric information. Metric information is essential in path planning specially to find the shortest path. Occupancy grid map itself provides the probability of occupancy and such information is valuable whenever robot operates in dynamic environment. Memory requirement and computational complexity for 2D occupancy grid are no more a huge burden for modern computers but for 3D occupancy grid is still challenging. Considering the above factors, feature based maps and occupancy

grid based maps are more common in SLAM. Focus on readily available map for path planning and obstacle avoidance, occupancy grid based maps are used in this thesis.

3.2 Scan Matching

3.2.1 Iterative Closest Point (ICP)

The iterative closest point (ICP) algorithm is widely used today for registration of 3D point clouds and polygonal meshes. ICP iteratively updates and refines the relative pose by minimizing the sum of squared distances between corresponding points in the two scans, and was introduced in 1992 in seminal papers on ICP are by P. J. Besl *et al* [45].

The first step of the algorithm is to find set of corresponding point pairs from two consecutive range scans. This search for point pairs from two range scan is the most time consuming step of the scan matching process. The naive way to do this is to pick up for each point its closest neighbor by Euclidean distance. While this will not in general be the correct corresponding point, especially if the scans are far apart from each other, successive iterations will still usually converge to a good solution.

It is necessary to assign different weights for point pairs since all of them may not correspond to same physical location. One may want to assign different weights to different point pairs, marking the confidence one has that they do indeed match. This can be done, for example, by setting the weight inversely proportional to the point to point distance and having lower weights for points further apart. For tunnel or corridor data, such linear weighting may degrade performance, as most of the points along the walls and ceiling will generally be well-aligned. In addition their influence will overwhelm point pairs with larger distances, which correspond to corners and

other features that are important. When registering scans with different amount of occlusions, taken from the same position, linear weighting is preferable, to reduce the effect of outliers and other non-overlapping points.

Some outlier pairs will also need to be rejected entirely. This can be done using some heuristic to reject pairs where the points are far apart. Points lying on surface boundaries should always be rejected. Otherwise, points from non-overlapping sections of the data may affect the accuracy of the final alignment. Since it is difficult to determine the boundary points for point cloud data, best solution is to decrease distance threshold for rejection instead. This way, pairs that are separated by a large distance are used in early iterations to bring the scans closer.In later iterations, point pairs which are far apart from each other will be rejected since they are likely to be from non-overlapping areas. The best choice of weighting and rejection strategies depends on the characteristics of the data. Finally, the measured distances between the point pairs are minimized and the process is repeated again, with a new selection of points, until the algorithm has converged.

For a pair of points: $P(x_i, y_i)$, $P'(x'_i, y'_i)$, $i = 1, ..., n$, a distance function between the transformed point set P and point set P' is defined as follows.

$$E_{dist}(\omega, T) = \sum_{i=1}^{n} |R_\omega P + T - P'|^2 \tag{3.1}$$

$$E_{dist}(\omega, T) = \sum_{i=1}^{n} ((x_i \cos\omega - y_i \sin\omega + T_x - x'_i)^2 + (x_i \sin\omega + y_i \cos\omega + T_y - y'_i)^2) \tag{3.2}$$

Where R_ω is rotation matrix and $T = [T_x, T_y]$. Closed-form solutions for T_x, T_y and ω can be calculated by minimizing $E_{dist}(\omega, T)$ as given below.

$$\omega = \arctan \frac{S_{xy'} - S_{yx'}}{S_{xx'} + S_{yy'}} \tag{3.3}$$

$$T_x = \bar{x}' - (\bar{x}\cos\omega - \bar{y}\sin\omega) \tag{3.4}$$

$$T_y = \bar{y}' - (\bar{x}\sin\omega + \bar{y}\cos\omega) \tag{3.5}$$

Where,

$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$,

$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$,

$\bar{x}' = \frac{1}{n}\sum_{i=1}^{n} x_i'$,

$\bar{y}' = \frac{1}{n}\sum_{i=1}^{n} y_i'$,

$S_{xx'} = \frac{1}{n}\sum_{i=1}^{n} (x_i - \bar{x})(x_i' - \bar{x}')$,

$S_{yy'} = \frac{1}{n}\sum_{i=1}^{n} (y_i - \bar{y})(y_i' - \bar{y}')$,

$S_{xy'} = \frac{1}{n}\sum_{i=1}^{n} (x_i - \bar{x})(y_i' - \bar{y}')$ and

$S_{yx'} = \frac{1}{n}\sum_{i=1}^{n} (y_i - \bar{y})(x_i' - \bar{x}')$.

The two largest problems with ICP are that, it is a point-based method that does not consider the local shape of the surface around each point and that the frequent nearest neighbor searches are computationally expensive.

3.3 Rao - Blackwellized Particle Filter for Grid Based SLAM

According to Murphy [38], key idea of Rao-Blackwellized particle filter for SLAM is to estimate the joint posterior $p(x_{1:t}, m|z_{1:t}, u_{1:t})$ about the map m and trajectory $x_{1:t} = x_1, ..., x_t$ of the robot. This estimation is performed given the control commands to the mobile robot $u_{1:t} = u_1, ..., u_t$ and the observations $z_{1:t} = z_1, ..., z_t$ obtained by the mobile robot. Rao-Blackwellized particle filter for SLAM makes use of the following factorization.

$$p(x_{1:t}, m|z_{1:t}, u_{1:t}) = p(m|x_{1:t}, z_{1:t}).p(x_{1:t}|z_{1:t}, u_{1:t}) \tag{3.6}$$

As a benefit of this factorization, trajectory of the robot can be estimated first and then to compute the map given that trajectory. Map only depends on the pose estimation of the robot and not the motion control commands to the robot. Thus, this approach offers efficient computation of the both robot trajectory and map.

Particle filter can be applied to estimate the posterior $p(x_{1:t}|z_{1:t}, u_{1:t})$ over the potential trajectories of the robot. Individual map is associated with each particle and they are updated by the observations and the trajectory represented by corresponding particle. Since $x_{1:t}$ and $z_{1:t}$ are known, the posterior $p(m|x_{1:t}, z_{1:t})$ can be computed analytically.

One of the most common particle filtering algorithm is sampling importance re-sampling (SIR) filter. A Rao-Blackwellized SIR filter for SLAM incrementally processes the sensor observations and the odometry readings as they are available. It updates the set of samples that represent the posterior of the robot trajectory and the map. This process has four steps, namely sampling, importance weighting,

re-sampling and map estimation. In sampling stage, the next generation of particles $\{x_t^{(i)}\}$ is obtained from the present generation of particles $\{x_t^{(i-1)}\}$ by sampling from the proposal distribution π. Often, probabilistic motion model is used as the proposal distribution. In next step individual importance weight $w_t^{(i)}$ is assigned to each particle according to the importance sampling principle.

$$w_t^{(i)} = \frac{p(x_{1:t}^{(i)}|z_{1:t}, u_{1:t})}{\pi(x_{1:t}^{(i)}|z_{1:t}, u_{1:t})} \tag{3.7}$$

The weights are assigned for the fact that the proposal distribution π is in general not equal to the target distribution. Particles are drawn with replacement proportional to their importance weights. After re-sampling all the particles will have the same weight. Final step of particle filter SLAM is to update the map. For each particle, the corresponding map estimate $p(m^{(i)}|x_{1:t}^{(i)}, z_{1:t})$ is computed based on the trajectory $x_{1:t}^{(i)}$ of that sample and the history of observations $z_{1:t}$. Weights of the trajectories should be evaluated from scratch whenever new observation is available. This procedure would be inefficient, since the length of the trajectory increases over time. Following assumption is used in order to overcome this inefficiency.

$$\pi(x_{1:t}|z_{1:t}, u_{1:t}) = \pi(x_t|x_{1:t-1}, z_{1:t}, u_{1:t}).\pi(x_{1:t-1}|z_{1:t-1}, u_{1:t-1}) \tag{3.8}$$

where π is the proposal distribustion.

Based on Eq. 3.7 and 3.8 weights are computed as,

$$w_t^{(i)} = \frac{p(x_{1:t}^{(i)}|z_{1:t}, u_{1:t})}{\pi(x_{1:t}^{(i)}|z_{1:t}, u_{1:t})} \tag{3.9}$$

$$= \frac{\eta p(z_t|x_{1:t}^{(i)}, z_{1:t-1}) p(x_t^{(i)}|x_{t-1}^{(i)}, u_t)}{\pi(x_t^{(i)}|x_{1:t-1}^{(i)}, z_{1:t}, u_{1:t})} \cdot \frac{p(x_{1:t-1}^{(i)}|z_{1:t-1}, u_{1:t-1})}{\pi(x_{1:t-1}^{(i)}|z_{1:t-1}, u_{1:t-1})} \tag{3.10}$$

$$\propto \frac{\eta p(z_t|m_{t-1}^{(i)}, x_t^{(i)}) p(x_t^{(i)}|x_{t-1}^{(i)}, u_t)}{\pi(x_t^{(i)}|x_{1:t-1}^{(i)}, z_{1:t}, u_{1:t})} \cdot w_{t-1}^{(i)} \tag{3.11}$$

where $\eta = 1/p(z_t|z_{1:t-1}, u_{1:t})$ is the normalization factor resulting from Bayes' rule and equal for all particles. Most of existing particle filter applications rely on the recursive structure of Eq. 3.11. Decision on how to calculate the proposal distribution and when to re-sample leaves open in this recursive formula. Samples are drawn from proposal distribution π in the prediction step in order to obtain the next generation of particles. The better the proposal distribution approximates the target distribution, the better is the performance of the particle filter. If it is possible to directly draw the samples from the target distribution, the importance weights would become equal for all particles and the re-sampling step would no longer be necessary. Unfortunately there is no way of draw samples directly from target distribution.

The typical particle filter applications use the odometry motion model $p(x_t|x_{t-1}, u_{t-1})$ as the proposal distribution. The motion model has the advantage that it is easy to compute. Importance weights are then calculated according to the observation model $p(z_t|m, x_t)$. This becomes clearly by replacing π in Eq. 3.11 by the motion model.

$$w_t^{(i)} = w_{t-1}^{(i)} \frac{\eta p(z_t|m_{t-1}^{(i)}, x_t^{(i)}) p(x_t^{(i)}|x_{t-1}^{(i)}, u_t)}{p(x_t^{(i)}|x_{t-1}^{(i)}, u_t)} \tag{3.12}$$

$$\propto w_{t-1}^{(i)} p(z_t|m_{t-1}^{(i)}, x_t^{(i)}) \tag{3.13}$$

However, this proposal distribution is suboptimal especially when the sensor information is preciser than the motion estimate of the robot based on odometry such as in

case of a robot equipped with a laser range finder. To overcome this problem, most recent observation z_t has to be considered when generating the next generation of samples as used in FastSLAM 2.0. Integration of z_t into proposal allows to focus the sampling on the meaningful regions of the observation likelihood. The distribution

$$p(x_t|m_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_t) = \frac{p(z_t|m_{t-1}^{(i)}, x_t)p(x_t|x_{t-1}^{(i)}, u_t)}{p(z_t|m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_t)} \tag{3.14}$$

is the optimal distribution with respect to the variance of the particle weights. With the integration of z_t into the proposal distribution, the computation of weights is as follows.

$$w_t^{(i)} = w_{t-1}^{(i)} \frac{\eta p(z_t|m_{t-1}^{(i)}, x_t^{(i)})p(x_t^{(i)}|x_{t-1}^{(i)}, u_t)}{p(x_t|m_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_t)} \tag{3.15}$$

$$\propto w_{t-1}^{(i)} \frac{\eta p(z_t|m_{t-1}^{(i)}, x_t^{(i)})p(x_t^{(i)}|x_{t-1}^{(i)}, u_t)}{\frac{p(z_t|m_{t-1}^{(i)}, x_t)p(x_t|x_{t-1}^{(i)}, u_t)}{p(z_t|m_t^{(i)}{}_1, x_{t-1}^{(i)}, u_t)}} \tag{3.16}$$

$$= w_{t-1}^{(i)} . p(z_t|m_{t-1}^{(i)}, x_{t-1}, u_t) \tag{3.17}$$

$$= w_{t-1}^{(i)} \int p(z_t|x')p(x'|x_{t-1}^{(i)}, u_t)dx' \tag{3.18}$$

Whenever modelling a mobile robot equipped with an accurate sensor like laser range finder, it is convenient to use an improved proposal distribution since accuracy of the laser range finder leads to extremely peaked likelihood functions.

In order to calculate the next generation of samples, first use a scan matching algorithm to determine the meaningful area of the observation likelihood function. Then sample that meaningful area and evaluate the sample points based on the

target distribution. For each particle i, the parameter $\mu_t^{(i)}$ and $\Sigma_t^{(i)}$ are determined individually for K sampled points $\{x_j\}$ in the meaningful area of the observation likelihood. Gaussian parameters are estimated as

$$\mu_t^{(i)} = \frac{1}{\eta^{(i)}} \sum_{j=1}^{K} x_j . p(z_t | m_{t-1}^{(i)}, x_j) \tag{3.19}$$

$$\Sigma_t^{(i)} = \frac{1}{\eta^{(i)}} \sum_{j=1}^{K} p(z_t | m_{t-1}^{(i)}, x_j)(x_j - \mu_t^{(i)})(x_j - \mu_t^{(i)})^T \tag{3.20}$$

with the normalization factor

$$\eta^{(i)} = \sum_{j=1}^{K} p(z_t | m_{t-1}^{(i)}, x_j) \tag{3.21}$$

In this way, closed form approximation of the optimal proposal can be obtained. Using this proposal distribution, the weights can be computed as,

$$w_t^{(i)} = w_{t-1}^{(i)} . p(z_t | m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_t) \tag{3.22}$$

$$= w_{t-1}^{(i)} \int p(z_t | m_{t-1}^{(i)}, x') dx' \tag{3.23}$$

$$\cong w_{t-1}^{(i)} \sum_{j=1}^{K} p(z_t | m_{t-1}^{(i)}, x_j) \tag{3.24}$$

$$= w_{t-1}^{(i)} . \eta^{(i)} \tag{3.25}$$

Where $\eta^{(i)}$ is the normalization factor that used in the computation of the Gaussian approximation of the proposal in Eq. 3.21.

3.3.1 Efficient Calculation of Proposal Distribution

Calculation of proposal distribution requires to evaluate Eq. 3.19 to Eq. 3.25. Efficient evaluation of measurement likelihood ($p(z_t|m_{t-1}^{(i)}, x_j)$) at the sampled points x_j in occupancy grid map is the key challenge. Likelihood fields for range finders algorithm in [46] is used to calculate $p(z_t|m_{t-1}^{(i)}, x_j)$.

3.3.2 Effective Sample Size for Selective Re-sampling

Re-sampling step has major influence on the performance of the particle filter. During re-sampling, particles with low importance weights $e^{(i)}$ are typically replaced by by samples with a high weight. But re-sampling is necessary since only a finite number of particles are used to approximate the target distribution. There is a risk of removing good particles from the filter which can lead to particle impoverishment and finally filter will diverge. To avoid that problem, A. Kong *et al* [39] and A. Doucet *et al* [47] introduced effective sample size to estimate how well the current particle represent the target posterior.

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} \left(\tilde{w}^{(i)}\right)^2} \tag{3.26}$$

Where $\tilde{w}^{(i)}$ refers to the normalized weight of particle i. If the samples were drawn from the target distribution, their importance weights would be equal to each other and N_{eff} equals to N where N is the number of particles. The worse the approximation of the target distribution, the higher is the variance of the importance weight and $0 \leq N_{eff} \leq N$ accordingly. Since N_{eff} can be regarded as a measure of the dispersion of the importance weights, it is useful to evaluate how well particle set approximates the target posterior and re-sampling is done whenever N_{eff} falls below a threshold value such as $0.75N$ or $0.5N$.

3.4 Loop Closing

SLAM problem is identified to overcome the drawback of inability to determine the robot pose only using odometry. Various SLAM techniques estimate the robot pose precisely but can not completely eliminate the error. Therefore residual error is accumulated over the motion of the robot. This can be clearly seen when robot entered to previously visited area after traversing a large loop. Due to accumulation of error, corresponding features detected at the beginning of the loop and at the end of the loop will not be at the same spatial location of the acquired map. Loop closing is performed in order to eliminate these topographical inconsistencies. Loop closing procedure contains three steps detection of loop closure, calculation of correction and distribute the correction over in the loop.

3.4.1 Detecting Loop Closure

Main step in loop closure is the detection of the opportunity to close a loop. Various techniques to detect the loop for different solutions SLAM problem are given in [3,5, 16,48,49]. The loop closure detection method used in this thesis is described here.

Each and every particle creates and updates a occupancy grid map $m^{(i)}$ and topological map $\mathcal{G}^{(i)}$ and both are updated while performing the SLAM. Vertices of the topological map $\mathcal{G}^{(i)}$, represent the positions visited by the robot and edges of $\mathcal{G}^{(i)}$ represent the trajectory corresponding to the ith particle. While the robot is progressing, nodes are added to $\mathcal{G}^{(i)}$, if distance to previous node exceeds a threshold or none of the other nodes is visible to current pose of the robot. Whenever, a new node is added to $\mathcal{G}^{(i)}$, an edge is also added from current node to most recently visited node. Ray casting operation in occupancy grid should be performed, in order to determine whether or not the previous node is visible from current pose of the robot or another

node other than previous node is visible to current node.

While robot is progressing and maps are continuously updated, if another node other than previous node is visible to the robot then there may be a possibility of closing a loop in the area. When the robot is further moving, if the distance to the newly visible node from the robot is decreasing in occupancy grid map $m^{(i)}$ while the distance to the same node is increasing in topological map $\mathcal{G}^{(i)}$, then it is confirmed that loop closing is imminent.

3.4.2 Calculation of Correction and Distribute over the Loop

In grid based SLAM the only available data is a set of points obtained from laser scans. Calculation of corrections to the poses along the loop can be treated as a scan matching problem. Instead of matching two scans, batch of scans at the beginning of the loop and at the end of the loop are matched in order to find the relative pose(accumulated residual error) of two batches of scans. The relative pose obtained from scan matching is the correction that is necessary to align corresponding features obtained at the beginning and at the end of the loop.

The accumulated residual error is distributed proportionally between all the poses along the loop to correct the accumulated error at the end of the loop. The error is distributed proportionally according to the magnitude of the displacement and turn at each motion.

3.4.3 SLAM Algorithm

SLAM algorithm used in this study is shown below, The algorithm requires the sample set of the previous time step($\mathcal{S}_{t-1}$), the most recent observation(z_t), the most recent control command to robot(u_{t-1}) and topological map for loop closing($\mathcal{G}$). Algorithm returns the new set of samples($\mathcal{S}_t$) and topological map for loop closing ($\mathcal{G}$)and correspond to current robot position.

for all $s_{t-1}^{(i)} \in \mathcal{S}$ **do**

$< x_{t-1}^{(i)}, w_{t-1}^{(i)}, m_{t-1}^{(i)} >= s_{t-1}^{(i)}$

//scan matching

$x_t'^{(i)} = x_{t-1}^{(i)} \oplus u_{t-1}$

$\hat{x}_t^{(i)} = argmax_x p(x|m_{t-1}^{(i)}, z_t, x_t'^{(i)})$

if $\hat{x}_t^{(i)} =$ failure **then**

$x_t^{(i)} \sim p(x_t|x_{t-1}^{(i)}, u_{t-1})$

$w_t^{(i)} = w_{t-1}^{(i)}.p(z_t|m_{t-1}^{(i)}, x_t^{(i)})$

else

//sample around the mode

for all $k = 1, ..., K$ **do**

$x_k \sim \{x_j | \, |x_j - \hat{x}^{(i)}| < \Delta\}$

end for

//compute Gaussian proposal

$\mu_t^{(i)} = (0, 0, 0)^T$

$\eta^{(i)} = 0$

for all $x_j \in \{x_1, ..., x_K\}$ **do**

$\mu_t^{(i)} = \mu_t^{(i)} + x_j.p(z_t|m_{t-1}^{(i)}, x_j)$

$\eta^{(i)} = \eta^{(i)} + p(z_t|m_{t-1}^{(i)}, x_j)$

end for

$\mu_t^{(i)} = \mu_t^{(i)} / \eta^{(i)}$

$\sum_t^{(i)} = 0$

for all $x_j \in \{x_1, ..., x_K\}$ **do**

$\sum_t^{(i)} = \sum_t^{(i)} + (x_j - \mu_t^{(i)})(x_j - \mu_t^{(i)})^T . p(z_t | m_{t-1}^{(i)}, x_j)$

end for

$\sum_t^{(i)} = \sum_t^{(i)} / \eta^{(i)}$

//sample new pose

$x_t^{(i)} \sim \mathcal{N}(\mu_t^{(i)}, \sum_t^{(i)})$

//update importance weight

$w_t^{(i)} = w_t^{(i)} . \eta^{(i)}$

end if

//update occupancy grid map

$m_t^{(i)} = \text{integrateScan}(m_{t-1}^{(i)}, x_t^{(i)}, z_t)$

//update sample set

$\mathcal{S}_t = \mathcal{S}_t \cup \left\{ < x_t^{(i)}, w_t^{(i)}, m_t^{(i)} > \right\}$

//update topological map

robotPosition = $(x_t^{(i)}.X, x_t^{(i)}.Y)$

dist = EucledianDistance(robotPosition, lastNode($\mathcal{G}^{(i)}$))

if dist $\geq 2m$ **then**

insertNewNodeTo($\mathcal{G}^{(i)}$)

flagNewNode = 1

end if

end for

$N_{eff} = \frac{1}{\sum_{i=1}^{N} \left(\tilde{w}^{(i)} \right)^2}$

if $N_{eff} < T$ **then**

$\mathcal{S}_t = resample(\mathcal{S}_t)$

end if

if flagNewNode == 1 **then**

hw = max(w_t)

visibleNode = rayCasting($\mathcal{G}^{(hw)}$)

if visibleNode == 1 **then**

visibleDistance = calculateVisibleDistance(visibleNode($\mathcal{G}^{(hw)}$), lastNode($\mathcal{G}^{(hw)}$))

distanceAlongGraph = calDistanceAlongGraph(visibleNode($\mathcal{G}^{(hw)}$), lastNode($\mathcal{G}^{(hw)}$))

if ($visibleDistance \leq 6m$) AND ($distanceAlongGraph \geq 20m$) **then**

doLoopCLosing

end if

end if

end if

3.5 Experiment

3.5.1 Mobile Robot Platform

An Active Media Pioneer 3 AT mobile robot equipped with SICK LMS 200 laser range finder is used to verify the validity of the algorithm. Pioneer 3 AT robot is equipped with onboard PC 104 computer and accessory cards. Robot has a differential drive system and each drive motor is equipped with quadrature optical shaft encoders for position and speed sensing. Robot is also equipped with a gyroscope to improve the estimation of robot position. SICK LMS 200 laser range finder is a two-dimensional scanning sensor. Device does not require any position marks or reflectors. Time of flight of laser pulse is used to estimate the range from sensor to the object. SICK LMS 200 has maximum range of 50m and angle resolution of 0.5^0 and 1^0. Angle resolution of 0.5^0 is used to have a better accuracy in scan matching.

3.5.2 Details of Mapped Area

Robot is traversed in the first floor corridor of S.J. Carew Building, Memorial University. Corridor is a rectangular loop about 65m in length and 28m in width. Raw odometry and lased range finder readings were extracted from the robot log. MATLAB program code is written to run the SLAM algorithm off-line.

3.5.3 Results

Figure 3.1 shows a map of the environment generated by raw odometry of the mobile robot. The map generated by odometry is not a complete map, it fails to map straight corridors and 90^0 corners correctly. Odometry errors are accumulated over the loop and as a result the generated map is erroneous.

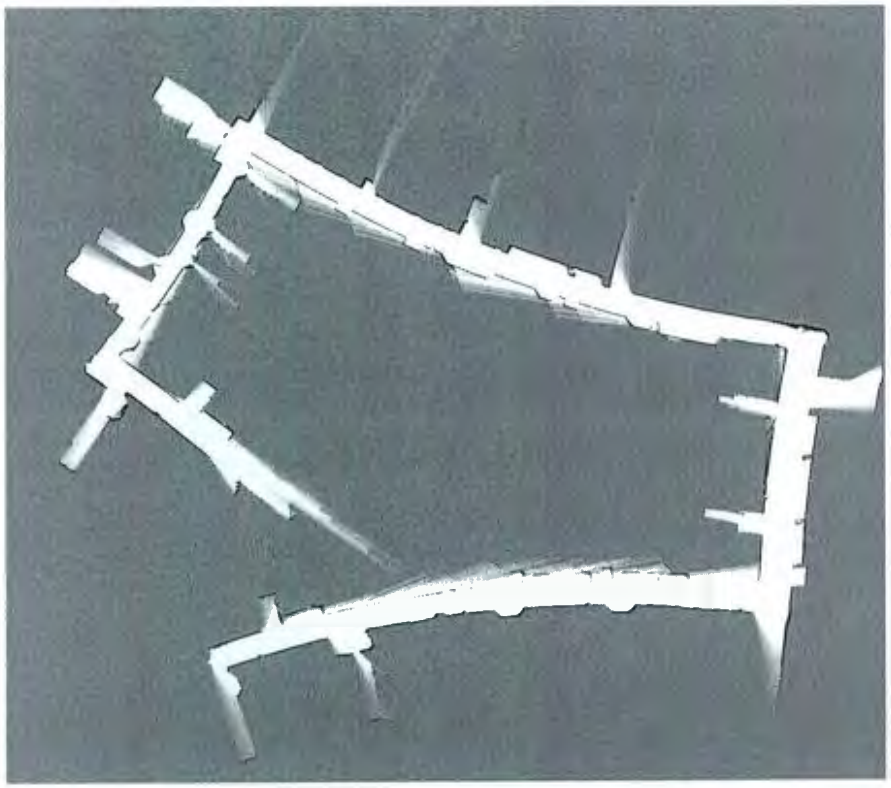

Figure 3.1: Map generated by raw odometry

The map in figure 3.2 was generated using laser stabilized odometry(scan matching). Compared to the map in figure 3.1, the map in figure 3.2 has better accuracy but still fails to generate an accurate map of the environment. There is a large error at the point of closing the loop. It can also be seen that, some sections of the mapped corridor are not straight while other sections are straight. As a result, the accumulated error along the loop will be non-linear. Error can not be compensated by calculation of the error and re-distributing along the loop. SLAM problem originally came to research community to compensate the problem of inability to build a map from either raw odometry or laser stabilized odometry(scan matching).

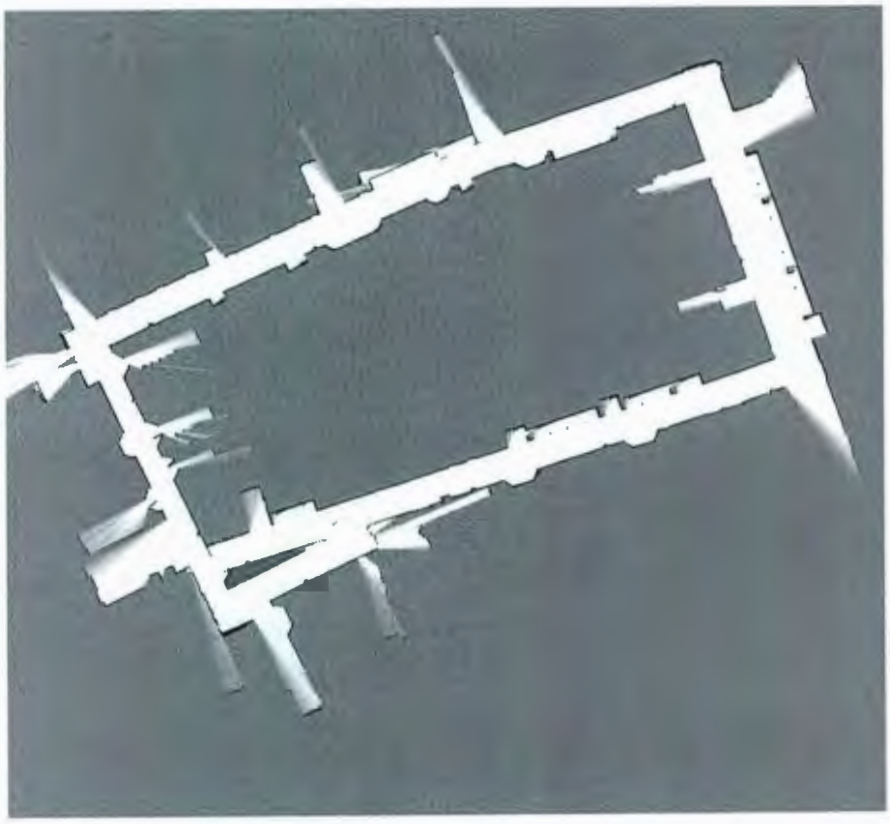

Figure 3.2: Map generated by scan matching

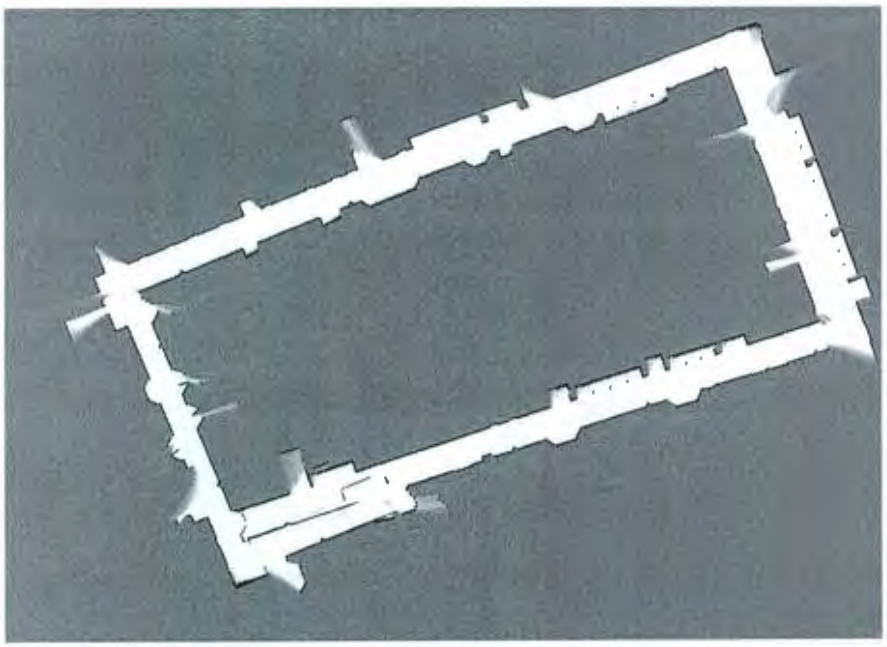

Figure 3.3: Map generated by SLAM algorithm and it is about to close a loop

Figure 3.3 shows the map generated by SLAM algorithm. There is an error at the point of loop closing. Except the error at the end of the loop, map is in good quality and it is not possible to observe any inconsistency in corridors of the map. Therefore it is possible to assume that the error at the end of the loop is an accumulation of residual errors of robot pose along the loop.

It is essential to calculate the correction necessary for the robot pose to regulate the error in the map. It can also be treated as a scan matching problem. Instead of matching two scans, batch of laser scans from the beginning and from the end of the loop is matched to estimate the error accumulated along the loop. Figure 3.4 shows the batch of 200 scans from the beginning of the loop and from the end of the loop. Points in red depict start of the loop and points in blue depict points from the end of the loop.

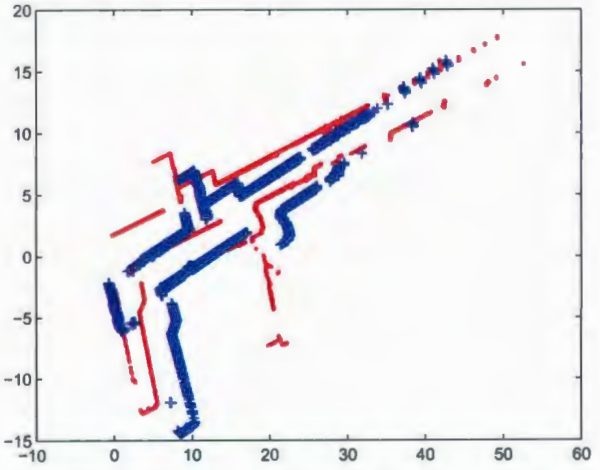

Figure 3.4: Selected points for loop closing from beginning of the loop and end of the loop. Points in red depict start of the loop and points in blue depict end of the loop

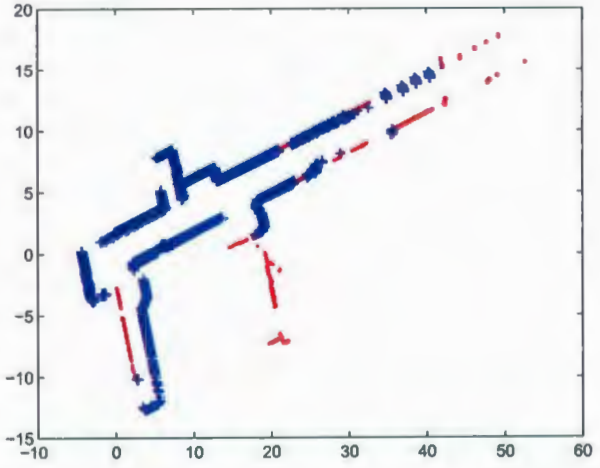

Figure 3.5: Proper alignment of two point sets after loop closing

Figure 3.5 shows proper alignment of two point sets after correcting the error due to accumulated error of robot pose along the loop.

Figure 3.6 shows the map of the environment after loop closing. It is the complete map of the environment that can be used for path planning and obstacle avoidance.

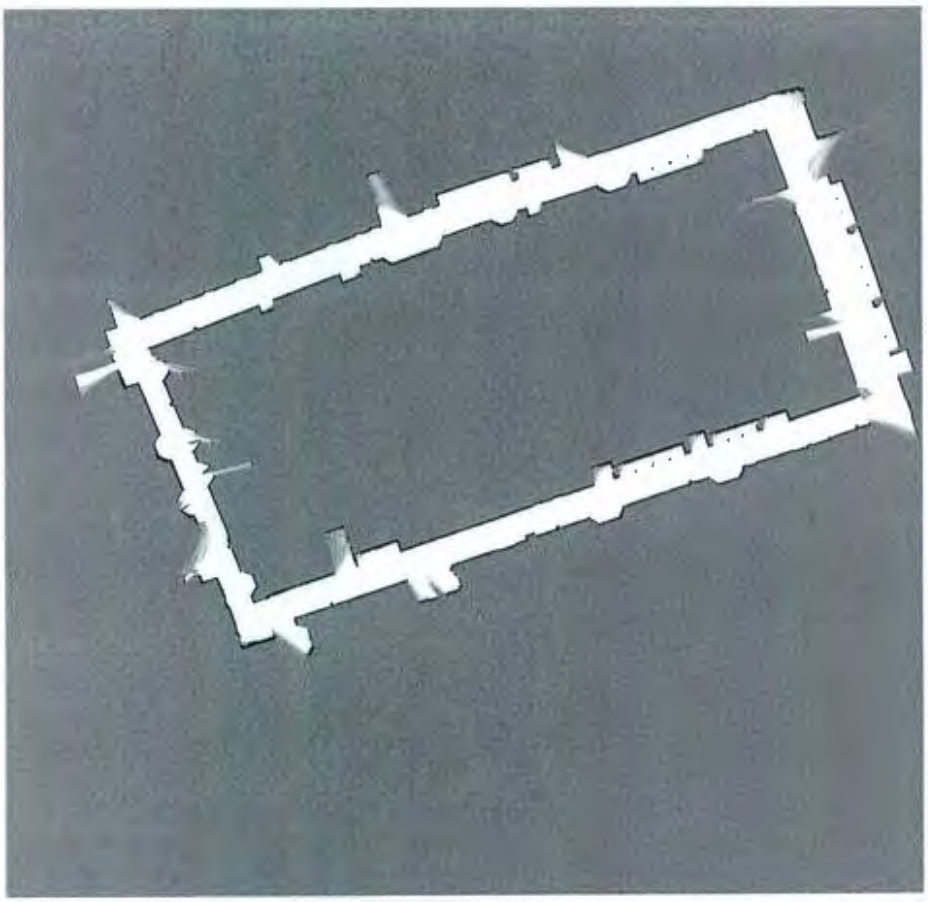

Figure 3.6: Map generated by SLAM algorithm after loop closing

The length of the building is 76 m and the width of the building is 28 m. Resolution of the map is 10 cm $\times$ 10 cm per grid cell and map is accurate according to the resolution of the map. There is no misalignments or structural inconsistencies hence map is consistent.

Chapter 4

Multi-Robot FastSLAM and Map Merging

4.1 Introduction

In multi-robot SLAM, two or more robots travel independently and provide individual maps. The important task is to merge all the maps from different robots, in order to have a complete map of the environment.Various solutions are presented in literature for joining maps from team of mobile robots.

G. Dedeoglu *et al* [50], S. Thrun *et al* [41] and S.B. Williams *et al* [22] proposed to localize a robot in another robot's map to find transformation between two maps. This method is computationally expensive. Markov localization makes the strong assumption that the robot is somewhere in the map and that can be easily lead to false location somewhere in the map (false positive).

W. H. Huang *et al* [51] proposed a method to merge topological maps. But the topological map mentioned in the algorithm is not the topological map generally discussed in robotics literature. Algorithm needs a specially created topological map

of wall like features obtained from a sensors such as laser range finder. Having such a map, algorithm searches for the possibility of aligning line like features of two maps. Results depict the proper merging of maps. The map necessary for the algorithm does not generated by any SLAM algorithm, therefore generating such a map will increase the computational burden. There will be less computational complexity, if it is possible to use image processing techniques to generate required topological map from occupancy grid map.

S. Carpin *et al* [52] address the map merging as an stochastic search to solve an optimization problem. The algorithm is based on adaptive random walks to merge occupancy grid maps independent from how they are generated. Suitable transformation aiming to overlap two grid maps is sought in the space of possible transformations. Further improvements are introduced by A. Birk *et al* [53] who attempts to detect failures and try to guide the search in order to obtain the result in an efficient manner. Algorithm is guaranteed to find optimal solution when the number of iterations tends to infinite. Computational requirement due to iterative nature of the algorithm prevents its use in real time operations.

X. Ma *et al* [54] introduced an adaptive genetic algorithm based method for occupancy grid map merging. Adaptive genetic algorithm is used to prevent premature convergence, low convergence rate and low stability. Simulation results confirm the feasibility of the algorithm. But simulation is done for a simple map. No results are presented for merging complex real world occupancy grid maps. Authors claim that the algorithm outperforms the adaptive random walk based map merging algorithm presented in [53]. But there is no any analysis of computational complexity and memory requirement to prove the efficiency of the algorithm.

S. Carpin *et al* [55] proposed novel algorithm to find out the transformation between two occupancy grid based maps using spectral information of the maps. Unlike

in [52], rotation is first calculated in this algorithm and then the corresponding translation is calculated. Several possible transformations are calculated in the algorithm. Authors introduced a simple calculation(acceptance index) to check how well two maps are aligned by each and every possible set of transformations. Acceptance index can have values between 0 and 1 but value will approach to 1 for correct transformation when there is good overlap. Transformations whose acceptance indexes are less than 0.9 should be discarded. This can happen either when maps do not overlap well to find reliable transformation between them or for invalid transformations. Finally the algorithm returns the correct transformation if such a transformation is possible. According to the authors experience only one will have acceptance index closer to 1 while others in the group will have acceptance indexes well below 0.9. This is not an iterative algorithm and solution will be available in single run enabling it to be used in real time.

4.2 Map Merging based on Hough Transform

Considering the map merging techniques discussed above, the Hough transform based technique provides accurate results. It allows to incorporate uncertainty and provides the best solution for map merging among set of candidate solutions. Adaptive random walk based and Hough transform based solutions rarely provide false positive since there is possibility to check the validity of map merging. They can reject all the candidate transformation if their acceptance indexes are less than the given threshold. According to S. Carpin *et al* [55], Hough transform based techniques are computationally efficient compared to adaptive random work technique based map merging. Therefore Hough transform based map merging technique used to merge map from different robot platforms.

4.2.1 Computation of Rotation between Two Maps

Orientation between two maps is calculated as first step of the algorithm. Occupancy grid map is transformed to a binary image by setting all occupied cells to black and all other cells to white. Then discrete Hough transform is applied. Discrete Hough transform discretizes the hough domain for ρ and θ and it can be represented by a matrix. On this point onwards, for a map M, the symbol $HT_{\mathcal{M}}$ is used to indicate the discrete Hough transform in this thesis. $HT_{\mathcal{M}}$ has θ_s columns and ρ_s rows. Associated hough spectrum for a given $HT_{\mathcal{M}}$ can be calculated as follows:

$$HS_{\mathcal{M}}(k) = \sum_{i=1}^{\rho_s} HT_{\mathcal{M}}(i,k)^2 \quad 1 \leq k \leq \theta_s. \tag{4.1}$$

Hough spectrum is extended periodically for values of k outside the range $1, ..., \theta_s$. Hough spectrum ($HS_{\mathcal{M}}$) is a measure of the directions where more frequent lines are detected in M. Hough spectrum is a unidimensional signal. Therefore cross correlation between two such signals can be used to determine similarities. Correlation outlines translations that will overlap two signals. Since Hough spectrum is defined over orientations, cross correlation should be calculated considering 2π periodicity into account. In other words, it is necessary to calculate circular cross correlation. For two hough spectrums $HS_{\mathcal{M}_1}$ and $HS_{\mathcal{M}_2}$ with same sampling periods, the circular cross correlation $CC_{\mathcal{M}_1\mathcal{M}_2}$ can be defined as follows:

$$CC_{\mathcal{M}_1\mathcal{M}_2} = \sum_{i=1}^{\theta_s} HS_{\mathcal{M}_1}(i) HS_{\mathcal{M}_2}(i+k) \quad 1 \leq k \leq \theta_s. \tag{4.2}$$

Cross correlation of Hough spectrums gives useful indications about how $HS_{\mathcal{M}_2}$ should be translated in order to overlap it with $HS_{\mathcal{M}_1}$. Translation of Hough spectrums corresponds to rotation of associated maps. Therefore local maxima of circular cross

correlation of Hough spectrums reveal how M_2 should be rotated in order to align it with M_1. Whenever circular cross correlation displays multiple local maxima, each of them associated with possible rotations. Instead of dealing with global maximum, the algorithm extracts n local maxima and n is the number of candidates for the possible transformation. Value for n can be defined at the beginning of the algorithm. Then the algorithm tracks all possible n transforms and find the best transformation as described later. Uncertainty can be added to the possible rotations for better accuracy of results. This can be done by introducing uncertainty (ϵ) to angles. Therefore for every angle θ obtained from the algorithm, there will be another two angles $\theta - \epsilon$ and $\theta + \epsilon$. Therefore all together there will be $3n$ candidate angles for possible rotations.

4.2.2 Computation of Translation between Two Maps

Once possible rotations are known, map M_2 can be rotated using candidate rotation angle i resulting map M_3. Spectral structures of two images M_1 and M_3 are extracted to calculate the translation between two maps. The X-spectrum of a binary image M is defined as follows:

$$SX_{\mathcal{M}}(j) = \begin{cases} \sum_{i=1}^{r} M(i,j) & 1 \leq j \leq c \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

Similarly Y-spectrum of image M is defined as follows:

$$SY_{\mathcal{M}}(j) = \begin{cases} \sum_{j=1}^{c} M(i,j) & 1 \leq i \leq r \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

where r and c are number of rows and number of columns respectively. $SX_{\mathcal{M}}$ and $SY_{\mathcal{M}}$ are the projections along the x and y axis of the image. Given the $SX_{\mathcal{M}_1}$ and

$SX_{\mathcal{M}_3}$, the translation along x axis is given by the global maximum of the cross correlation between them. It is defined as follows:

$$CCX_{\mathcal{M}_1\mathcal{M}_3}(\tau) = \sum_{k=-\infty}^{+\infty} SX_{\mathcal{M}_1}(k+\tau)SX_{\mathcal{M}_3}(k) \tag{4.5}$$

Translation along y axis can be calculated in similar way using following equation.

$$CCY_{\mathcal{M}_1\mathcal{M}_3}(\tau) = \sum_{k=-\infty}^{+\infty} SY_{\mathcal{M}_1}(k+\tau)SY_{\mathcal{M}_3}(k) \tag{4.6}$$

At the end of this step, all together there will be n possible transformations. Acceptance index(ω)is calculated in order to find the best transformation from candidate transformations. Acceptance index is a measure accuracy of candidate transformations.

4.2.3 Acceptance Index

Let M_1 and M_2 be two maps with r rows and c columns. The agreement between M_1 and M_2 represented by $arg(M_1, M_2)$ is the number of overlapping cells in M_1 and M_2 after transformation that are either both free or both occupied. The disagreement between M_1 and M_2 represented by $dis(M_1, M_2)$ is the number of overlapping cells after transformation such that M_1 is free and M_2 is occupied or vice-versa. The acceptance index between two maps is defined as

$$\omega(M_1, M_2) = \begin{cases} 0 & if\ arg(M_1, M_2) = 0 \\ \frac{\arg(M_1,M_2)}{\arg(M_1,M_2)+dis(M_1,M_2)} & if\ arg(M_1, M_2) \neq 0 \end{cases} \tag{4.7}$$

Only occupied and unoccupied cells are considered to calculate the acceptance index while simply ignoring the unknown cells. Two extremes are reached when the maps

do not agree in a single cell or when they are the same. When two maps do not agree in a single cell, $arg(M_1, M_2)$ will be zero hence ω is zero. When two maps are identical $dis(M_1, M_2)$ is zero, hence ω is one. Values between these two boundaries are obtained for intermediate situations. According to [52], acceptance index values less than 0.9 indicate either the two maps does not overlap well or there is no suitable transformations and candidate transformation should be discarded.

4.3 Experiment

Laser and odometry readings from the experiment in Chapter 3 were taken and sensor measurements are divided in to two parts with overlap. Set of readings from the end of the data set were omitted to have one set of readings and set of readings from beginning of the readings were omitted to generate another set of readings. Two readings sets were treated as they were obtained from two different robots. Maps obtained from two sets of readings are shown in Figure 4.1 and Figure 4.2.

Hough spectrums of map 1 and map 2 are shown in figure 4.3 and figure 4.4 respectively. Circular cross correlation of two Hough spectrums are in figure 4.5. Each and every local maximum of circular cross correlation is a candidate angle for possible rotation between two maps. Since there is no rotation between them there is only one maximum at 1^0. By incorporating uncertainty ϵ of one degree which use throughout the experiment it will have the exact angle 0^0.

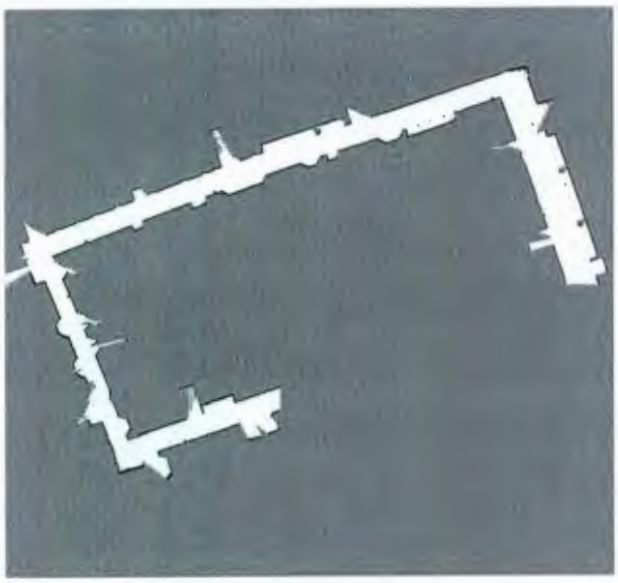

Figure 4.1: Map 1 for map merging

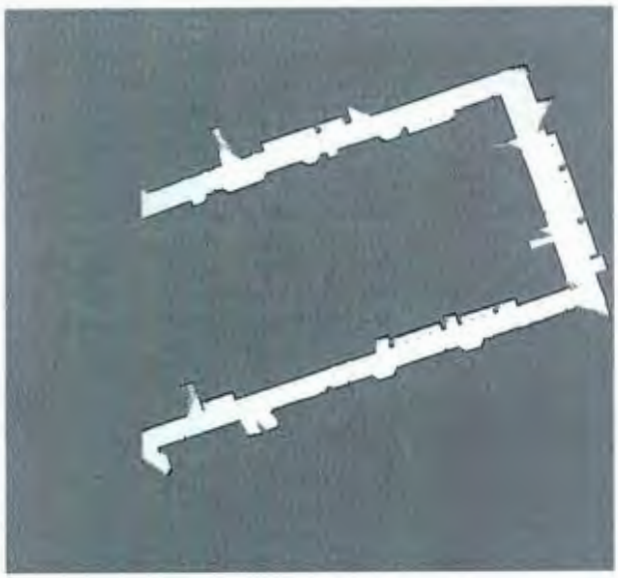

Figure 4.2: Map 2 for map merging

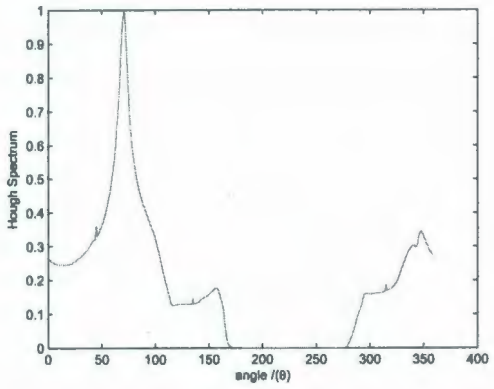

Figure 4.3: Hough Spectrum of Map 1

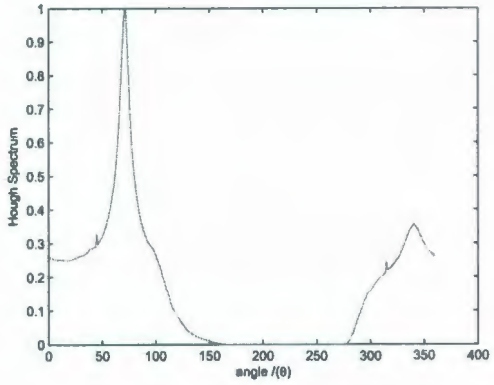

Figure 4.4: Hough Spectrum of Map 2

Having candidates for possible rotations, the next step of the algorithm is to find all the possible transformations. Map 2 has to be rotated for each and every candidate angle for rotation leading to have calculation of large number of X-spectrums and Y-spectrums. Figures of X-spectrums and Y-spectrums whose correspond to final successful transformation are shown in this thesis. X-Spectrums of map 1 and map 2 are shown in figure 4.6 and figure 4.7 respectively. Cross correlation between them is shown in figure 4.8. Y-Spectrums of map 1 and map 2 are shown in figure 4.9 and figure 4.10 respectively. Cross correlation between them is shown in figure 4.11. Two

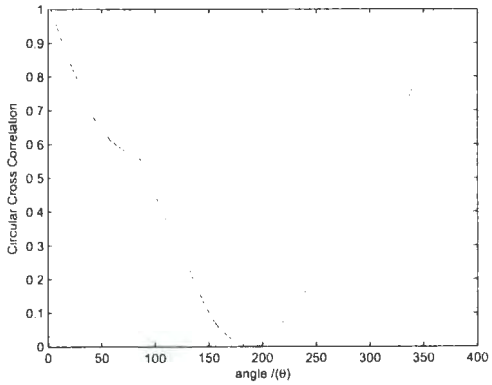

Figure 4.5: Circular cross correlation of Hough Spectrums of two maps

maps are treated as images hough transform based map merging. Therefore required translation in both X-direction and Y-direction are shown in image coordinates.

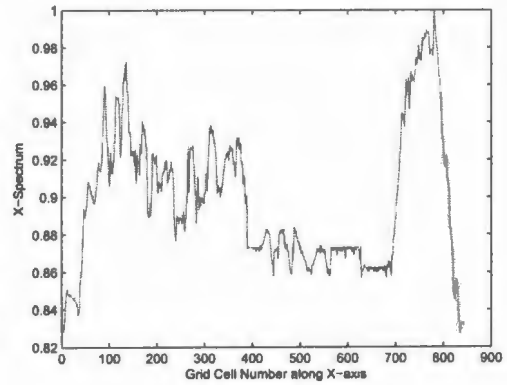

Figure 4.6: X-Spectrums of map 1

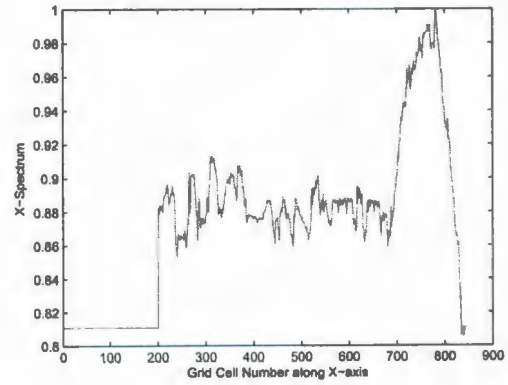

Figure 4.7: X-Spectrums of rotated map 2

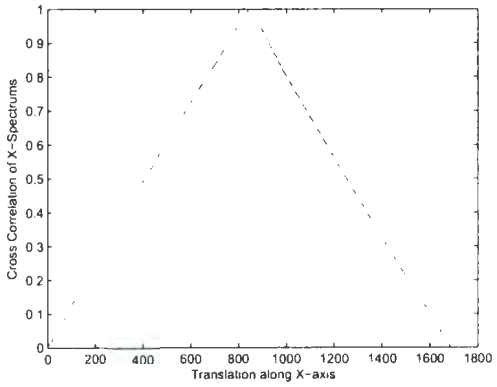

Figure 4.8: Cross correlation of X-spectrums of two maps

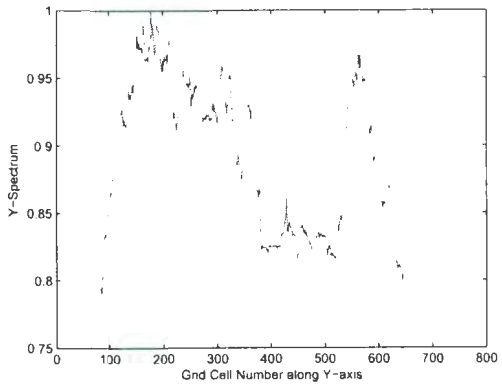

Figure 4.9: Y-Spectrums of map 1

Merged map is shown in figure 4.12 and it has the same accuracy compared to map generated from single robot SLAM.

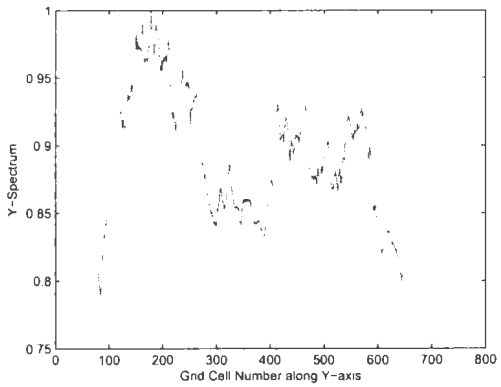

Figure 4.10: Y-Spectrums of rotated map 2

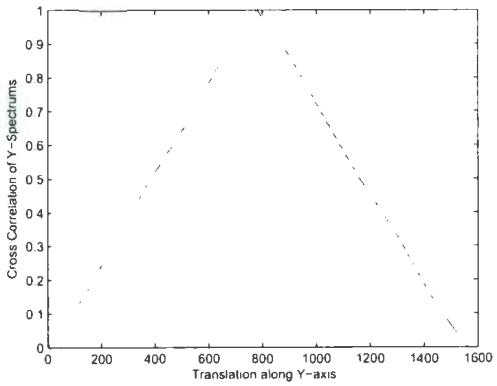

Figure 4.11: Cross correlation of Y-spectrums of two maps

$\omega(M_1, M_2)$ for map in Fig 4.12 is 0.9774. If $\omega(M_1, M_2)$ is less than 0.9 those transform should be discarded. This can be happen either when there is no enough overlap between two maps or when there is no relationship between two maps in real world. Map obtain from multi robot SLAM is in good quality compared to map obtained from single robot SLAM. But map obtained from single robot SLAM is more accurate since loop closing is applied to that map. F. Lu *et al* [56] introduced to avoid the problem due to non loop closed section in multi robot SLAM. It worked well since map is simple but this method will fail when there is non loop closed

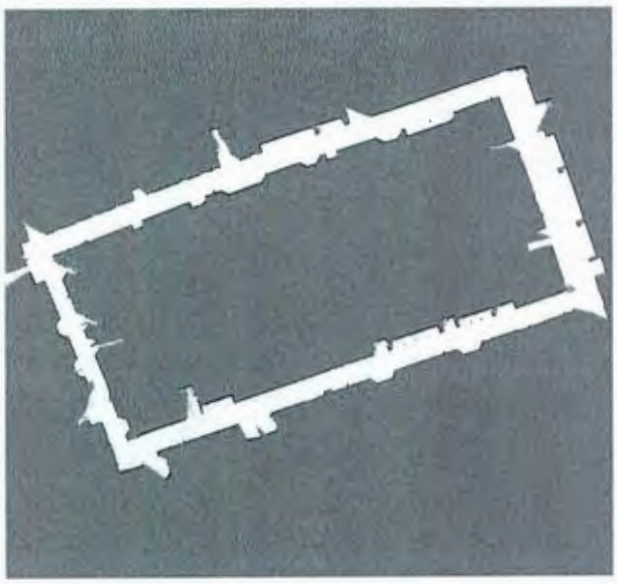

Figure 4.12: Complete map of the environment

complex maps. Best way to overcome this problem is to maintain list of mutual observation among other robots in the team(robot rendezvous) and apply graph based optimization techniques to apply constrains found during mutual observations. This method is mentioned in future works of this thesis. Computational time for map building in single robot SLAM example is less since there is overlapping area in multi robot SLAM. Significant area of the maps is falls into overlapping area and this cause the higher computational time. If there were maps with enough overlapping for map merging and the overlapping area is a less percentage of the total map area, then there will be less time for multi robot SLAM. Time taken to loop closing is sam as the time taken to map merging and algorithm in [56]. Robot power consumption is higher in single robot SLAM and there was low battery warning when complete the loop. Travelling distance for single robot is less in multi robot SLAM and there will not be problem of low battery even if two single robots were introduced for terrain

acquisition. Considering the power consumption by single robot and multiple robot, computational complexity and capability of robots, multi robot system is efficient to deploy and the larger the higher the efficiency.

Chapter 5

Conclusion

5.1 Conclusion

FastSLAM offers computationally efficient algorithm to solve SLAM problem compared with other available solutions. EKF based SLAM is the earliest and widely used solution to SLAM problem in early days. But it suffers from higher computational complexity and memory requirement and this burden limits the size of the environment that can be mapped using EKF-SLAM. FastSLAM obtains the path of the robot from particle filter, hence calculation of landmark location will be independent of each other. Employing particle filter to estimate the path of the robot makes it easier to estimate landmark locations. Reduced computational complexity enables to map larger environments. FastSLAM allows to use an occupancy grid map instead of landmark list while EKF-SLAM can only use list of landmarks. Path planing and obstacle avoidance become easy with occupancy grid maps. However, when an area becomes larger multi-robot SLAM is more efficient and also has the advantage of sharing the computational burden among several robots. Solving SLAM problem using multiple robot is important when there is large terrain to map and perhaps it

will beyond the capability of single robot. Even if it is within the capability of single robot, such a deployment will not be cost effective and will not be time effective.

Odometry and sensor readings from Pioneer 3AT were divided in to two sets with overlap. Single robot SLAM algorithm described in Chapter 3 was applied to both data sets. Two partial maps were obtained by treating as if two data sets were obtained from two robots. Map merging technique described in Chapter 4 is employed to combine two maps. Resulting map of the map merging algorithm has the same accuracy as the map of the environment generated by single robot SLAM. Results prove the feasibility of employing multiple robots which divide the computational burden of map building and terrain acquisition among team of robots.

5.2 Future Works

Although proposed algorithm performs well, there are modifications to the algorithm that will improve the robustness of the algorithm. Hough spectrum based map merging algorithm does not allow to incorporate previous knowledge of relative orientation between two maps, even if they are available. This situation can arise when two robots mutually recognize each other during mapping. This drawback can be eliminated by checking line to line structural consistency. Hough transform based line detection can be employed to extract lines from two occupancy grid maps.

There may be situations that two robots will mutually recognize each other more than once at different locations. In such situations all these conditions should be considered when merge two maps from those two robot. Hough spectrum based techniques may fail because still there may be residual errors due to loop closing. Best way to deal with this problem is to maintain a list of mutual observation and employ graph based optimization technique to merge maps. This will help the algorithm

to close larger loops which single robot in the group has not explored that loop completely.

Bibliography

[1] S. Thrun. Robotic mapping: A survey. February 2002.

[2] C. Stachniss, D. Hahnel, and W. Burgard. Exploration with active loop-closing for fastslam. In *International Conference on Intelligent Robots and Systems, 2004. (IROS 2004).*, pages 1505 1510, October 2004.

[3] J. Folkesson and H. I. Christensen. Closing the loop with graphical slam. *IEEE Transactions on Robotics*, 23(4):731 – 741, August 2007.

[4] L. Morenoa, S. Garridoa, D. Blancoa, and M. L. Muñozb. Differential evolution solution to the slam problem. *Robotics and Autonomous Systems*, 57(4):441 450, April 2009.

[5] B. Williams, M. Cummins, J. Neira, P. M. Newman, I. D. Reid, and J. D. Tardós. A comparison of loop closing techniques in monocular slam. *Robotics and Autonomous Systems*, 57(12):1188 1197, 2009.

[6] P.W. Gibbens, G.M.W.M.Dissanayake, and H.F. Durrant-Whyte. A closed form solution to the single degree of freedom simultaneous localisation and map building (slam) problem. In *IEEE Conference on Decision and Control*, Sydney NSW, Australia, December 2000.

[7] M.W.M.G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building problem. *IEEE Transactions of Robotics and Automation*, 17(3):229–241, June 2001.

[8] M. Montemerlo, S. Thurn, D. Koller, and B. Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. *AAAI 2002 Conference Proceedings*, pages 593–598, 2002.

[9] M. Montemerlo and S. Thurn. Simultaneous localization and mapping with unknown data association using fastslam. *ICRA 03*, 2:1985 – 1991, September 2003.

[10] M. Montemerlo, S. Thurn, D. Koller, and B. Wegbreit. Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003. IJCAI.

[11] D. Hahnel, W. Burgard, D. Fox, and S. Thrun. An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *International Conference on Intelligent Robots and Systems, 2003. (IROS 2003).*, pages 206 – 211, October 2003.

[12] A. Eliazar and R. Parr. Dp-slam: Fast, robust simultaneous localization and mapping without predetermined landmarks. In *In Proc. 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-03)*, pages 1135 – 1142. Morgan Kaufmann, 2003.

[13] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *International Journal of Robotics Research*, 23(7 - 8):693 – 716, August 2004.

[14] A. Eliazar and R. Parr. Dp-slam 2.0. In *In Proceedings IEEE International Conference on Robotics and Automation(ICRA), 2004*, pages 1314 1320, Apr 26-May 1 2004.

[15] M. Bosse, P. Newman, J. Leonard, and S. Teller. Simultaneous localization and map building in large-scale cyclic environments using the atlas framework. *Int. J. Robotics Research,*, 23(12):1113 1140, 2004.

[16] C. Stachniss, D. Haehnel, W. Burgard, and G. Grisetti. On actively closing loops in grid-based fastslam. *Advanced Robotics - The Int. Journal of the Robotics Society of Japan (RSJ)*, 19(10):1059 – 1080, 2005.

[17] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23:34 46, 2007.

[18] S. Grzonka, C. Plagemann, G. Grisetti, and W. Burgard. Look-ahead proposals for robust grid-based slam with rao-blackwellized particle filters. *International Journal of Robotics Research*, 28(2):191 – 200, 2009.

[19] S. Thrun and Y. Liu. Multi-robot slam with sparse extended information filers. In *Proceedings of the 11th International Symposium of Robotics Research (ISRR'03)*, pages 254 – 266, Sienna, Italy, 2003. Springer.

[20] Z. Wang, S. Huang, and G. Dissanayake. Multi-robot simultaneous localization and mapping using d-slam framework. In *3rd International Conference on Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007*, pages 317 322, December 2007.

[21] X.S. Zhou and S.I. Roumeliotis. Multi-robot slam with unknown initial correspondence: The robot rendezvous case. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006*, pages 1785–1792, October 2006.

[22] S.B. Williams, G. Dissanayake, and H. Durrant-Whyte. Towards multi-vehicle simultaneous localisation and mapping. In *Proceedings IEEE International Conference on Robotics and Automation, ICRA '02*, pages 2743–2748, May 2002.

[23] E.W. Nettleton, H.F. Durrant-Whyte, P.W. Gibbens, and A.H. Goktogan. Multiple platform localisation and map building. In *Sensor Fusion and Decentralized Control in Robotic Systems III*, November 2000.

[24] E.W. Nettleton, P.W. Gibbens, and H. F. Durrant-Whyte. Closed form solutions to the multiple platform simultaneous localisation and map building (slam) problem. In *AeroSense*, April 2000.

[25] C. Estrada, J. Neira, and J.D. Tardos. Finding good cycle constraints for large scale multi-robot slam. In *Proceedings of IEEE International Conference on Robotics and Automation, ICRA '09, 2009.*, May 2009.

[26] E.W. Nettleton, Sebastian Thrun, Hugh Durrant-Whyte, and S. Sukkarieh. Decentralised slam with low-bandwidth communication for teams of vehicles. In *International Conference on Field And Service Robotics (FSR 2003)*, Yamanashi, Japan, July 2003.

[27] A. Howard. Multi-robot simultaneous localization and mapping using particle filters. *International Journal of Robotics Research*, 25(12):1243–1256, 2006.

[28] A. Howard, G.S. Sukhatme, and M.J. Mataric. Multirobot simultaneous localization and mapping using manifold representations. In *Proceedings of the IEEE*, pages 1360-1369, July 2006.

[29] R. Smith and P. Cheeseman. On the representation of spatial uncertainty. *Int. J. Robotics Research*, 5(4), 1987.

[30] R. Smith, M. Self, and P. Cheeseman. A stochastic map for uncertain spatial relationships. In *on The fourth international symposium*, pages 467 474, Cambridge, MA, USA, 1988. MIT Press.

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

[32] S. Thrun, W. Burgard, and D. Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning and Autonomous Robots(joint issue)*, 31(1):29 - 53, 1998.

[33] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping (slam): part 1 the essential algorithms. *IEEE Robotics and Automation Magazine*, 13(2):99 - 110, June 2006.

[34] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46 - 57, Jun 1989.

[35] D. Amarasinghe. *Multisensor Based Environment Modelling and Control Applications for Mobile Robots.* PhD thesis, Memorial University of Newfoundland, July 2008.

[36] Momotaz Begum. Robotic mapping using soft computing methodologies. Master's thesis, Memorial University of Newfoundland, July 2005.

[37] M. Montemerlo and S. Thurn. *FastSLAM A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics.* Springer, 2007.

[38] K. P. Murphy. Bayesian map learning in dynamic environments. In *Advances in Neural Information Processing Systems(NIPS)*, pages 1015 1021. MIT Press, 1999.

[39] A. Kong, J. S. Liu, and W. H. Wong. Sequential imputations and bayesian missing data problems. *Journal of the American Statistical Association*, 89(425):278 288, March 1994.

[40] A. Howard. Multi-robot simultaneous localization and mapping using particle filters. In *Robotics Science and Systems*, pages 201 208, Jun 2005.

[41] S. Thrun. A probabilistic on-line mapping algorithm for team of mobile robots. *International Journal of Robotics Research*, 20(5):335 - 363, 2001.

[42] Dirk Hähnel. *Mapping with mobile robots.* PhD thesis, University of Freiburg, December 2004.

[43] R. Siegwart and I. R. Nourbakhsh. *Introduction to Autonomous Mobile Robots.* MIT Press, 2004.

[44] H. P. Moravec and A. Efes. High resolution maps from wide angle sonar. In *In Proceedings IEEE International Conference on Robotics and Automation*, pages 116 121, March 1985.

[45] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239 – 256, February 1992.

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

[47] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice.* Springer, 2001.

[48] M. Kaess and F. Dellaert. A markov chain monte carlo approach to closing the loop in slam. In *International Conference on Robotics and Automation*, pages 643 – 648, Barcelona, Spain, April 2005.

[49] K. L. Ho and P. Newman. Loop closure detection in slam by combining visual and spatial appearance. *Robotics and Autonomous Systems*, 54:740 – 749, 2006.

[50] G. Dedeoglu and G. Sukhatme. Landmark-based matching algorithm for cooperative mapping by autonomous robots. In *DISTRIBUTED AUTONOMOUS ROBOTICS SYSTEMS*, pages 251–260. Springer-Verlag, 2000.

[51] W. H. Huang and K. R. Beevers. Topological map merging. *International Journal of Robotics Research*, 24(8):601–613, August 2005.

[52] S. Carpin, A. Birk, and V. Jucikas. On map merging. *Robotics and Autonomous Systems*, 53:1 – 14, August 2005.

[53] A. Birk and S. Carpin. Merging occupancy grid maps from multiple robots. *IEEE: Special Issue on Multi-Robot Systems*, 94(7):1384 – 1397, July 2006.

[54] X. Ma, R. Guo, Y. Li, and W. Chen. Adaptive genetic algorithm for occupancy grid maps merging. In *Proceedings of the 7th World Congress on Intelligent Control and Automation*, pages 5716 – 5720, Chongqing, China, June 2008.

[55] S. Carpin. Fast and accurate map merging for multi-robot systems. *Autonomous Robots*, 25:305 – 316, 2008.

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