MAP MERGING FOR MULTI ROBOT SLAM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ORHAN KARADENİZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

DECEMBER 2014

Approval of the thesis:

MAP MERGING FOR MULTI ROBOT SLAM

submitted by **ORHAN KARADENİZ** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Gönül Turhan Sayan _____
Head of Department, **Electrical and Electronics Engineering**

Assoc. Prof. Dr. İlkay Ulusoy _____
Supervisor, **Electrical and Electronics Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Aydan Erkmen _____
Electrical and Electronics Engineering Dept., METU

Assoc. Prof. Dr. İlkay Ulusoy _____
Electrical and Electronics Engineering Dept., METU

Prof. Dr. Gözde Bozdağı Akar _____
Electrical and Electronics Engineering Dept., METU

Assoc. Prof. Dr. Afşar Saranlı _____
Electrical and Electronics Engineering Dept., METU

Onur Alper Erdener _____
Director, ASELSAN

Date: 16.12.2014

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Orhan Karadeniz

Signature :

ABSTRACT

MAP MERGING FOR MULTI ROBOT SLAM

Karadeniz, Orhan

MS., Department of Electrical and Electronics Engineering

Supervisor : Assoc. Prof. Dr. İlkay Ulusoy

December 2014, 170 pages

In the area of mobile robotics Simultaneous Localization and Mapping (SLAM) is a challenging problem. In the literature, there are many solutions to this problem for single robots. However, multi-robot SLAM is a relatively new topic, which has additional issues, such as communication, task sharing and map merging. This thesis takes map merging as its focus and this is examined in terms of the specifications for the unknown initial positions of robots. In the map-merging scenario, every robot localizes itself and generates maps individually and the generated local maps of each robot are shared with other robots. This information

sharing can be achieved within different architectures. A distributed approach is used in the study reported in this thesis. This approach does not need a fully connected communication network and a central unit to accumulate the information.

This thesis examines the map-merging problem of multi robot SLAM though different approaches in literature. The single robot SLAM problem is solved with Compressed Extended Kalman Filter. The challenging part of map merging is the problem of the unknown initial position is solved with map similarity algorithms, the Delaunay Triangulation and triangle similarity metric. The stochastic search of global transformation matrix is undertaken applying the Random Sample Consensus, which is used for estimation of the transformation between the individual maps created by the robots. In the final step, the overlapping regions of the transferred maps are merged with different algorithms such as Maximum Likelihood, M-Estimator and Covariance Intersection. For experimental purposes, an open code simulator for the multi robot SLAM is implemented. Finally, each algorithm is examined under different scenarios and their performance analyses in relation to simulated and real world datasets, are presented in Chapter 5, which contains the details of the experiments.

Keywords: Map Merging, Multi Robot, SLAM, Map Similarity, Global Map Transformation

ÖZ

ÇOKLU ROBOTLAR İÇİN HARİTA BİRLEŞTİRME

Karadeniz, Orhan
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. İlkay Ulusoy

Aralık 2014, 170 sayfa

Eş Zamanlı Konumlandırma ve Haritalama (SLAM) problemi robotik alanının zorlu bir problemidir. Literatürde, bu sorunun tek robot için birçok çözümü bulunmatadır. Fakat, çoklu robot SLAM nispeten daha yeni bir konudur. Bu problem iletişim, görev paylaşımı ve harita birleştirme gibi ekstra konuları içermektedir. Harita birleştirme bu tezin ana konusudur, ve bilinmeyen ilk robot pozisyonları kısıtı altında incelenmiştir. Bu senaryoda her robot kendini konumlandırır ve haritalama yapar. Sonrasında oluşan haritalar diğer robotlar ile paylaşılır. Bu paylaşım farklı mimariler ile gerçeklenebilir. Bu çalışmada tam bağlı komünikasyon ağı ve merkezi bir işleme birimine ihtiyaç duymayan, dağınık mimari kullanılmıştır.

Bu tezde, çoklu robot uygulamalarında literatürdeki farklı harita birleştime yöntemleri detaylı incelenmiştir. Tek robot SLAM problemi Sıkıştırılmış Genişletilmiş Kalman Filtre (CEKF) algoritması ile çözülmüştür. Dağınık mimarilerde harita birleştimenin zor bir bölümü olan bilinmeyen ilk robot pozisyonları sorunu haritalar arası benzerlik bulap, ve bu benzerlik üzerinden haritalar arasındaki dönüşüm matrisi bulunarak çözülmüştür. Delanuay Üçgenleme yöntemi harita yapısından geometrik özellikler çıkararak, ve bu üçgenler üzerinden benzerlik hesaplanarak, haritalar arası benzer alanlar çıkarılmıştır. Bu benzer alanlar kullanılarak dönüşüm matrisi RANSAC algoritması kullanılarak aranmıştır. Son olarak haritalar bulunan matris ile aynı koordinat düzlemine taşınmış ve örtüşen bölgelerideki aynı sınır işaretleri Maximum Likelihood, Değiştirilmiş M-Tahminleyici ve Kovaryans Kesişimi (CI) algorithmaları ile birleştirilmiştir. Açık kaynak kodlu çoklu robot SLAM simülatörü gerçeklenmiş ve deneylerde kullanılmıştır. Son olarak, algoritmalar similasyon ve gerçek harita verileri kullanılarak değişik senaryolar ile incelenmiş, ve performans analizleri deneyler ve çıkarımlar bölümünde sunulmuştur.

Anahtar Kelimeler: Harita Birleştirme, Çoklu Robot, Harita Benzerliği

To My Dear Wife

ACKNOWLEDGEMENTS

I would like to thank to my supervisor Assoc. Prof. Dr. İlkay Ulusoy for her guidance, criticism and insight throughout the research.

I am also grateful to ASELSAN Inc. for encouragements and resources that are supported for this thesis.

I owe my deepest gratitude to my precious wife for her encouragement and continuous support, and my family for their trust and support.

TABLE OF CONTENTS

ABSTRACT v
ÖZ vii
TABLE OF CONTENTS xi
LIST OF TABLES xv
LIST OF FIGURES xvi
CHAPTERS 1
1. INTRODUCTION 1
1.1. Problem Definition and Motivation 1
1.2. Scope of the Thesis 5
1.3. Outline of the Thesis 6
2. LITERATURE SURVEY 7
2.1. Map Representation 7
2.2. Simultaneous Localization and Mapping 10
2.2.1. Feature Extraction 13
2.2.2. Data Association 14
2.3. Multi-Robot SLAM 17
2.3.1. Global Transformation 22
2.3.1.1. Global Transformation Using the Relative Measurements 23
2.3.1.2. Global Transformation Using the Map Overlap 24
2.3.1.2.1. Map Structure Similarity 26
2.3.1.2.2. Multi-Robot Data Association 29
2.3.2. Map-Merging 32
2.4. Evaluation of the Map Performance 38

3. THEORETICAL BACKGROUND 43
3.1. Data Association 43
3.1.1. Individual Compatibility Nearest Neighbor 43
3.1.2. Joint Compatibility Branch and Bound 45
3.2. Filters for SLAM Purpose 46
3.2.1. Kalman Filter 46
3.2.2. Extended Kalman Filter 48
3.2.3. Compressed Extended Kalman Filters 50
3.3. Global Map Transformation 55
3.3.1. Global Map Transformation by Using Relative Measurements 55
3.3.1.1. Relative Distance and Bearing Measurements 56
3.3.1.2. Transformation between Global Frames 57
3.3.2. Global Map Transformation by Using Map Overlap 60
3.3.2.1. Map Structure Similarity 60
3.3.2.1.1. Delaunay Triangulation 60
3.3.2.1.2. Similarity Metric Calculation 64
3.3.2.2. Multi Robot Data Association 66
3.3.2.2.1. Transformation Matrix Formulation 66
3.3.2.2.2. Random Sample Consensus 67
3.3.2.2.3. Adaptive Random Walk 73
3.3.2.2.4. Iterative Translation Search 75
3.4. Feature Based Map Merging 75
3.4.1. Maximum Likelihood Estimator 75
3.4.2. Modified M-Estimator 79
3.4.3. Covariance Intersection Estimator 80
3.4.4. Orthogonal Gnanadesikan-Kettenring Estimator 83
3.5. Map Performance Evaluation 84
3.5.1. Normalized Estimation Error Squared Consistency Test 84

3.5.2. Area of Covariance Ellipse Metric 85
4. SIMULATOR AND MAPS 87
4.1. Simulation Environment 87
4.1.1. Random Map Generator 92
4.1.2. Trajectory Planning 93
4.1.3. Motion Model 95
4.1.4. Observation Model 97
4.1.4.1. Bounding Box Test 98
4.1.4.2. Bounding Line Test 99
4.1.4.3. Bounding Circle Test 99
4.2. Maps and Trajectories 101
4.2.1. Extraction of Different Landmark Datasets 101
4.2.1.1. Obtaining Meaningful Real World Map 102
4.2.1.2. Manual Landmark Labeling 103
4.2.2. Extraction of Different Trajectories 106
4.2.2.1. Obtaining Meaningful Trajectories 106
4.2.2.2. Manual Trajectory Planning 107
5. EXPERIMENTAL RESULTS 109
5.1. Performance Analysis Experiments 111
5.1.1. Global Map Transformation Algorithms 111
5.1.1.1. Delaunay Triangulation 112
5.1.1.2. Similarity Metric Calculation 113
5.1.1.2.1. Computational Cost 113
5.1.1.2.2. Success Rate 114
5.1.1.3. Random Sample Consensus 117
5.1.1.3.1. Computational Cost of Algorithm Steps 117
5.1.1.3.2. Success Rate under Noisy Input 118
5.1.1.3.3. Success Rate under a Mismatched Noisy Input 119

5.1.1.3.4. Success Rate under Mismatched Noisy Input and RST .. 122
5.1.1.3.5. Computational Cost under Mismatched Noisy Input 123
5.1.1.3.6. Success Rate under Different Distance Methods 125
5.1.1.3.7. Computational Cost under Different Distance Methods . 126
5.1.2. Map Merging Algorithms 127
5.1.2.1. Computational Cost of Algorithms 128
5.1.2.2. Algorithm Parameter Optimization 130
5.1.2.2.1. Covariance Intersection Sensitivity Analysis 130
5.2. Simulated Map and Trajectory Experiments 133
5.2.1. Sensor Bearing Noise Effect on Estimator Performance 134
5.2.2. Robot Velocity Effect on Estimator Performance 137
5.2.3. Trajectory and Sensor Bearing Effect on Map Merging Performance 139
5.2.4. Trajectory and Sensor Range Effect on Map Merging Performance 142
5.3. Real World Map and Trajectory Experiments 145
5.3.1. Sensor Bearing Noise Effect on Map Merging Performance 145
5.3.2. Robot Velocity Noise Effect on Map Merging Performance 148
5.3.3. Sensor Range Effect on the Map Merging Performance 150
5.4. Robustness Analysis of Applied Algorithms 152
6. CONCLUSION AND FUTURE WORK 155
REFERENCES 163

LIST OF TABLES

TABLES

Table 1 Pseudo Code of Individual Compatibility Nearest Neighbor 43
Table 2 Pseudo Code of Bowyer-Watson's Incremental Delaunay Triangulation Algorithm [67]..... 63
Table 3 Pseudo Code of Adaptive Random Walk Algorithm [11] 74
Table 4 Pseudo Code of Iterative Translation Search [45] 76
Table 5 Pseudo Code of Orthogonal Gnanadesikan-Kettenring Estimator [53]... 83
Table 6 Computational Cost of Covariance Intersection Algorithm..... 129
Table 7 The NEES values of Purposed Algorithms..... 153

LIST OF FIGURES

FIGURES

Figure 1 Illustration of a Feature-Based Map and the Robot Trajectory [9]..... 8
Figure 2 Illustration of Topological Map [10] 8
Figure 3 Illustration of a Grid-Based Map [11] 9
Figure 4 Mono Dimensional Spurious Measurement Scenario [30]..... 16
Figure 5 Illustration of Different Algorithm Behaviors under the Same Scenario [30] 17
Figure 6 Illustration of the Effect of Transformation on the Overlapping Area ... 25
Figure 7 The upper two maps explored by different robots are merged into the lower resultant map [11]..... 33
Figure 8 Ellipses with a larger area are the landmark covariance matrices in different maps, which are merged with the smaller (inner) ellipses. 34
Figure 9 State Cycle Diagram of Kalman Filter..... 46
Figure 10 Illustration of Active and Stable Regions of Map 51
Figure 11 Relationship Between Positions and Measurements of Robots [42] 57
Figure 12 Geometric Relationship Between Global Frames and Landmark Position [42] 59
Figure 13 Demonstrations of Possible Triangle Shapes..... 61
Figure 14 Possible Outputs of DT under Exceptional Case (Rectangular Shape) 61
Figure 15 Illustration of Relationship between Voronoi Diagram and Delaunay Triangulation 62
Figure 16 Illustration of Similar Triangles with Different Landmarks 65
Figure 17 RANSAC Flow Chart 70
Figure 18 Geometric Illustration of Covariance Matrices [47]..... 82
Figure 19 Simulator Environment Screenshot 88
Figure 20 Multi Robot Simulator Environment Screenshot..... 89
Figure 21 Map Dimensions Popup Dialog Box 89

Figure 22 Manual Landmark Entry View 90
Figure 23 Saving Window View 90
Figure 24 Manuel Trajectory Planning Ability 91
Figure 25 Trajectory Saving Ability 91
Figure 26 Illustration of Generated Maps with and without Distance Limitation (without Distance Limitation and with 10 meters Distance Limitation) 93
Figure 27 Flow Chart of the Trajectory Planning Algorithm 94
Figure 28 Rotational Motion of Robot 95
Figure 29 Picture of Pioneer3-AT Robot 96
Figure 30 Picture of LMS 200 97
Figure 31 Illustration of the Scan Model of LMS 200 on Top View 98
Figure 32 Bounding Box Test Illustration 98
Figure 33 Bounding Line Test Illustration 99
Figure 34 Bounding Circle Test Illustration 100
Figure 35 Victoria Park with Labeled Landmarks 103
Figure 36 The 2 Dimensional Map of Central Park [86] 104
Figure 37 The 3 Dimensional Map of Central Park [86] 104
Figure 38 The 2 Dimensional Map of Central Park with Labeled Trees 105
Figure 39 Manual Landmark Entered Landmarks of Central Park 105
Figure 40 Victoria Park Trajectories 107
Figure 41 Central Park Trajectories 108
Figure 42 Multi-Robot Map-Merging Flow Diagram 110
Figure 43 Sample Size Effect on Delaunay Triangulation Performance 112
Figure 44 The Effect of Sample Size on the Similarity Calculation Performance 114
Figure 45 Illustration of Overlap Ratio Change 115
Figure 46 Effect of Overlap Ratio Change on Similarity Calculation Performance 116
Figure 47 Time Cost Illustration of Hypothesis and Test Steps 118

Figure 48 The Success Rate of RANSAC 119
Figure 49 Success Rate of RANSAC for Different Match Ratios 120
Figure 50 Success Rate of RANSAC under Different Match Ratios 123
Figure 51 Computational Cost of RANSAC under Different Match Ratios 124
Figure 52 Success rate of RANSAC using Different Distance Calculation Methods 125
Figure 53 Computational Cost of RANSAC under Different Distance Calculation Methods 127
Figure 54 Computational Cost of Map Merging Algorithms 128
Figure 55 Simulation Scenario of Sensitivity Analysis of Covariance Intersection 131
Figure 56 Sensor Bearing Noise Effect on the Consistent Landmark Number ... 132
Figure 57 Sensor Bearing Noise Effect on Determinant 132
Figure 58 Covariance Estimations of Map Merging Algorithms 133
Figure 59 Simulated Map and Robot Trajectories 134
Figure 60 Sensor Bearing Noise Effect on RANSAC Performance 135
Figure 61 Sensor Bearing Noise Effect on the Consistent Landmark Number ... 136
Figure 62 Sensor Bearing Noise Effect on the Determinant 136
Figure 63 Robot Velocity Noise Effect on RANSAC Performance 138
Figure 64 Robot Velocity Noise Effect on the number of Consistent Landmarks 138
Figure 65 Robot Velocity Noise Effect on the Determinant 139
Figure 66 Simulated Map and Trajectory Illustration 140
Figure 67 Robot Velocity Noise Effect on Map Similarity and RANSAC Performance 141
Figure 68 Sensor Bearing Noise Effect on the Consistent Landmark Ratio 141
Figure 69 Sensor Bearing Noise Effect on the Determinant 142
Figure 70 Simulated Map and Trajectory Illustration 143
Figure 71 Sensor Range Effect on Map Similarity and RANSAC Performance 143

Figure 72 Sensor Range Effect on the Consistent Landmark Ratio 145
Figure 73 Sensor Range Effect on the Determinant 145
Figure 74 The Victoria Park Extracted Map and Trajectory Illustration 146
Figure 75 Sensor Bearing Noise Effect on Map Similarity and RANSAC Performance 147
Figure 76 Sensor Bearing Noise Effect on Consistent Landmark Ratio 148
Figure 77 Robot Velocity Noise Effect on Map Similarity and the RANSAC Performance 149
Figure 78 Robot Velocity Noise Effect on NEES 149
Figure 79 The Victoria Park Extracted Map and Trajectory Illustration 150
Figure 80 Sensor Range Effect on Map Similarity and the RANSAC Performance 151
Figure 81 Sensor Range Effect on NEES 152
Figure 82 The Victoria Park Map and Trajectory Illustration 153

CHAPTER 1

INTRODUCTION

1.1. Problem Definition and Motivation

Robots are being increasingly used in many application areas, where the situation would be difficult, dangerous or life threatening for human beings; for example, natural and man-made disasters. Robots are also used in industrial contexts, particularly in the space industry. However, robots also face difficulties, which include determining the environment they are in and mapping that environment. The capacities of a robot are based on their physical structure and the extent of their ability to communicate. In an unknown area, the robot needs to localize itself and explore the environment. The sensor types and algorithms used for solving the problem localization and mapping diverge In terms of whether the environment is indoor or outdoor.

The interior of buildings and tunnels are examples of indoor environments. Parks, roads, underwater or air platforms are examples of the outdoor environment. There have been many robot applications in these type of environments, such as indoors [1] [2] [3] [4], and outdoors [5] [6] [7]. The indoor environments are small areas containing features with concentrated distribution. The outdoor environments consist of larger areas with fewer features. Since features are the objects, which can be scanned by the robots' sensors the feature density of the environment is one of the main determinants for the selection of the type of sensor type. In outdoor environments features are more rare, so more precise information about the location of features is needed and it is this information that is used for robot localization and mapping.

The sensors used by a robot can be divided into types; bearing-only, and bearing and range types. Monocular cameras, and sonic scanning and ranging sensors are examples of the bearing-only sensor type, which only give information about direction of the feature. Light-based ranging sensors (LIDAR) are an example of the bearing and range sensor type, providing information about not only direction but also distance between the features. Despite being more expensive, bearing and range sensor types provide more precise information than the bearing-only sensor types.

Using the information about the environment obtained through the sensors, a robot can solve the localization and mapping problem, which is called Simultaneous Localization and Mapping (SLAM) problem in the robotics area. For a robot, the SLAM problem is hard to solve due to the complexity of simultaneously extracting the map and localizing its own position.

Different algorithms can be used to solve the SLAM problem. However, despite the precision of sensors, there are still errors in sensor readings. Moreover, the imprecise nature of the SLAM problem results in the stochastic representation of the robot's position and map representation. As a result, with their ability to address complex and stochastic problems, probabilistic algorithms, such as Kalman Filter and Particle Filter, are more suitable for the SLAM problem. Both these algorithms have certain advantages under different conditions. For example, the Kalman Filter is an optimal estimation method when linear Gaussian noise models with zero mean are used; however, in nonlinear and different noise distributions, the Particle Filter is more advantageous. Therefore, in order to choose the right algorithm, the problem should be analyzed in detail. In this study, the Kalman Filter was used to solve the SLAM problem. The main reason for choosing the Kalman Filter was its optimal estimation property. Moreover, the Kalman Filter provides the feasibility of using the Gaussian distribution for the assumption of the features, and linearizing the model using the Taylor Series expansion method.

Filters are used for the estimation of the robot's position and the description of the environment. Environment representation is called a map representation. The map representation of an environment can be performed using feature-based or grid-based maps.

A feature-based map representation consists of the extracted properties of features. For example, a tree in an environment is represented by x and y coordinates. On the other hand, in a grid-based map representation, the map is divided into cells. Each evenly spaced cell has a binary random variable, which represents the probability of an obstacle to occur in that cell. The grid-based map representation has high computational and storage requirements compared with the feature-based representation.

Although algorithms have been successfully used for the solution of the single-robot SLAM problem, the multi robot case is still unresolved. This is due to the issues of task assignment, communication topology and map-merging in multi-robot applications.

Task assignment is employed when multiple robots are used to explore the same environment for an investigation. Task assignment decreases the time required for exploring the whole investigation area. However, in this study, task assignment was not investigated, and the tasks of the robots were assumed to be initially assigned.

A communication topology is the structure of communication between robots. There are different topologies; such as fully connected, and partially connected. The fully connected communication topology uses extra devices to overcome the problems of limited bandwidth and limited communication range. The bandwidth of a network is the rate of data transfer, and it is measured in bits per second. Limited communication range is the range limitation of the data transfer devices. Extra devices transfer the information from one robot to all other robots. On the other hand, in partially connected systems, the bandwidth and range limitation problems are solved through one to one communication between the robots. In

these systems, robots only use the information obtained from other robots that are within the communication range. The fully and partially connected systems of communication are also referred to as centralized and decentralized systems, relatively.

Centralized systems require one central unit for collecting and processing the accumulated information [8]. These requirements can be supplied by the fully connected communication topology. On the other hand, decentralized systems do not require a central unit, nor a fully connected communication topology. In decentralized systems, every robot is capable of using the information obtained from any other robot without having to obtain information from all other robots. In this study, decentralized system requirements were one of the main system specifications. A detailed description of the centralized and decentralized systems is given in Section 2.3.

Map-merging is one of the tasks to be performed to solve the multi-robot SLAM. In this task, the shared information is used to improve the precision of the global map. A global map is a cumulative map, which is the combination of individual maps of all robots. Two different scenarios can be used to examine the map-merging process. In the first scenario, every robot knows the relative positions of other robots. Therefore, each robot updates its location and builds a map based on this information. In the second scenario, the robots consider their initial positions as the origin of frame and construct their maps based on this assumption, which results in different frames for all robots. In the known initial position case, when the communication between robots is established, the combination of the local maps of robots is trivial. Therefore, there is no need for a map transformation calculation since the positions are already known. On the other hand, since the initial positions and relative frames are unknown in the second case, before combining the local maps of robots, frame transformation needs to be performed. As a result, the second case is more challenging than the first case. A detailed description of the unknown initial position scenario is given in Section 2.3.1.

After the global transformation, two maps in the same global frame are obtained for the construction of the final map. These two maps can have overlapping regions, which can be used to increase the precision of the map. This is called the process of combining maps. In this process, probability distributions of features are used as the inputs and the combinations of these distributions are given as the outputs of the process. These outputs with non-overlapping regions form the final global map. A detailed description of the combining maps process is given in Section 2.3.2.

In summary, despite the more complex issues involved, multi-robot SLAM applications have the advantage of providing a faster and more precise exploration of the environment. Moreover, they have a fault tolerance for failures resulting from robots and algorithms. Environmental conditions or design mistakes can damage the electronic or mechanic parts of robots. Moreover, algorithm failures can results in mistakes in localization and mapping, such as incorrect data association and wrong model assumptions. These failures can be overcome using multiple robots. In addition, the required time decreases with the increase in the number of robots. Moreover, the overlapping regions of the maps provide precise information, since the information is obtained from different robots. Therefore, the multi robot SLAM applications have attracted more interest than single-robot SLAM applications in recent years.

1.2. Scope of the Thesis

The scope of this study was to provide a map-merging algorithm for a team of robots. Each robot localized and mapped the unknown environment without being aware of the initial positions of other robots. The inputs for all robots were; sensor readings (bearing and range) and odometer data (velocity and angular velocity). Each robot takes measurement without affecting each other, which means correlation between them is not available. The output was the merged global map of the environment.

The inputs were used for the single-robot SLAM purpose. The Compressed Extended Kalman Filter was used to estimate the robot pose and construct a map of the environment. Each robot shared its local map with the other robots that were within the communication range. The similar regions of these shared maps were found using the Delaunay Triangulation algorithm to extract the geometric information from the map; such as the circumference and area of triangles... These similar regions were used to perform the global map transformation between the local maps of the robots. Then, the best global map transformation was found using the Random Sample Consensus algorithm to align the maps. The overlapping features in the aligned maps were merged through a different algorithm, involving the use of Maximum Likelihood Estimation, Modified M-Estimator and Covariance Intersection Estimator.

1.3. Outline of the Thesis

The structure of the thesis is given below.

Chapter 2 presents a survey of previous research conducted on the multi-robot SLAM problem and map merging process.

Chapter 3gives the theoretical background and formulation of the related literature under the following five sections, single-robot data association, filters for SLAM, global map transformation, feature-based map merging and map performance evaluation.

In Chapter 4, simulator and maps used in experiments are represented in two sections; simulation environment, and maps and trajectories.

In Chapter 5, experiments and results regarding the examined algorithms are given in the following sections; performance and sensitivity analysis experiments, simulated map and trajectory experiments and real world map and trajectory experiments.

Chapter 6 presents a summary of the research conducted within the scope of this thesis, the contributions to the literature and suggestions for future work.

CHAPTER 2

LITERATURE SURVEY

In this part of the thesis, a literature survey on the probabilistic SLAM algorithms, map representations, and the multi-robot SLAM problem are presented. The detailed results of the research are given in the following sections.

2.1. Map Representation

The environment representation is called a map representation. The selection of the map representation is important in the SLAM algorithms. The map representations of an environment can be mainly divided into two; feature-based and grid-based maps.

In feature-based map representation, a mean vector and a variance matrix are obtained from each extracted feature and stored separately. Figure 1 illustrates a feature-based map representation with the robot trajectory to clarify the stochastic nature of the SLAM algorithm.

In Figure 1, k represents the steps, x_k indicates the robot's state in step k and $z_{k,j}$ shows the measurement of the j^{th} feature on the map at the step k. The robot's estimation of its own state and the landmarks on the map is not the same as the ground truth, which is a good simulation of the map representation and the SLAM problem.

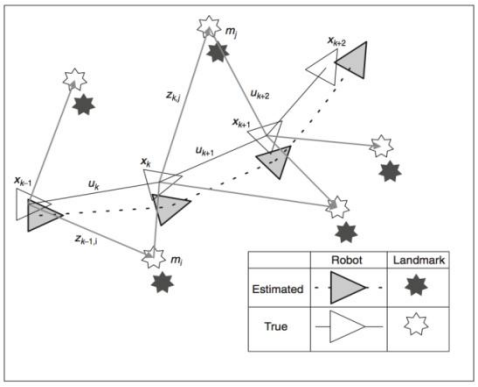


Figure 1 Illustration of a Feature-Based Map and the Robot Trajectory [9]

Topological maps are feature-based maps that contain additional information, such as the degree of vertices and orientations of edges at vertices in a graphical structure [10]. These vertexes are the nodes representing the positions of the robot and features. The edges connect one vertex to another using the distance between features. Figure 2 presents an example for the topological map representation.

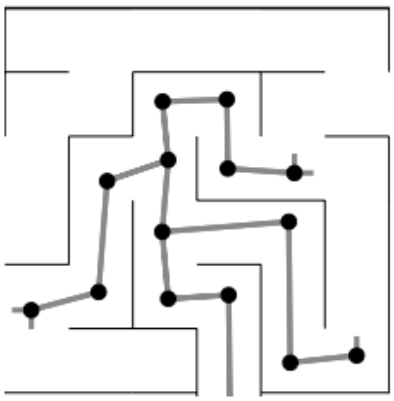


Figure 2 Illustration of Topological Map [10]

In grid-based map representations, the environment is represented as an evenly spaced field of binary random variables, which represent the probability of obstacle to occur in a given cell [11]. Grid-map representations require a high storage capacity and greater computation time for larger maps, but no landmark extraction algorithms or data association are needed. An example of grid-based map representation is given in Figure 3.

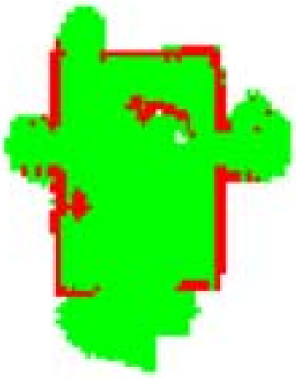


Figure 3 Illustration of a Grid-Based Map [11]

A graph-based map can be considered a sub-representation method for grid-based map representation [12]. In this representation, the trajectory with related scan measurements is used for map representation. Similarly, appearance-based map representation approaches [13][14] can also be considered graph-based map representations. In these approaches, snapshots of environment, captured by the camera on the robot, are used. The appearance-based map representation has the advantage of detecting the previously visited areas since this method uses visual features of the environment.

Different map representations can be chosen for different areas according to their suitability. For instance, for large-scale areas with predefined landmarks, the feature-based map representation is more appropriate than the grid-based map representation. On the other hand, the latter can be used for an unconstructed environment with a denser structure [15].

There is interesting research on map representations, investigating the combined use of feature and grid-based map representations. Sim [16] and Ho [17]used this combination for the SLAM algorithm. However, Sim's method only uses grid map for navigation and feature map for pose estimation, and Ho's method uses feature map for only loop closure and grid map for pose estimation. Another important research in this area is the one by Wurm [18],who used both map representations simultaneously for the estimation of the robot pose.

2.2. Simultaneous Localization and Mapping

SLAM is the localization and mapping problem of building a map of the unknown environment features while simultaneously finding the trajectory, which is also known as concurrent mapping and localization [19][20]. The main issue in this problem is the unknown environment and the unknown position of the robot, which correlates with the map and the robot trajectory. Different studies have investigated and developed methods to localize the known environments and map the known localizations [21]. However, SLAM is far more complicated than these problems due to its simultaneous nature. The SLAM problem is explained in detail in Section 1.1, in which, different methods used in the previous studies to solve the SLAM problem are also given.

In the robotics area, probabilistic SLAM solutions attract more attention than the deterministic solutions, with their more complex natures. Probabilistic solutions also require approximation to address the impossibility of considering the entire probability space. In addition to approximation, a discrete model for the motion of the robot is also required to decrease the high computational cost of updating the motion. Therefore, probabilistic approaches are needed to approximate the best solution and create discrete motion models. There is a crucial relation between the model dependency and the precision of a solution. If a solution relies more on the model, then the computational cost is less. With the computational gain, the precision of the solution decreases. In fact, the model derived from the problem itself is uncertain. Therefore, when the solution depends on the model, it is more likely to be wrong. If the solution does not highly rely on the model, the solution is more likely to perform better in real world implementations, but the computational cost increases. There are two main filter classes used to solve this stochastic problem, Particle Filter and Kalman Filter.

In Particle Filter, the zero mean Gaussian noise is not assumed; therefore, a linear system model is not needed. Since this method uses samples of distribution, its performance relies on the sample count. Under the smoothness assumptions, the

posterior probability tends to converge the correct value when the sample count goes to infinity [1]. Therefore, the sample count and the performance requirements of the system must be optimized in Particle Filter applications. However, using large sample sets for Particle Filter is not appropriate under real time specifications.

The Kalman Filter is an alternative solution to the SLAM problem, which is also the optimal solution for certain assumptions. It is commonly used in SLAM applications due to its optimality. The assumptions of the Kalman Filter are; Markov assumption, linear state and measurement transition assumption, independent zero mean and Gaussian system noise assumption, and the assumption based on the uniform initial distribution of the robot. A detailed explanation of the Kalman Filter and its assumptions is given in Section 3.1, and the optimality proof can be found in Thrun's book [1]. Since it is difficult to satisfy these assumptions in real world applications, improvements have been made to the original Kalman Filter resulting in the development of the Extended Kalman Filter, Unscented Kalman Filter, Compressed Kalman Filter and Covariance Intersection. For instance, the Extended Kalman Filter uses the Taylor series expansion to overcome linear system assumptions by linearizing the nonlinear system models around the current mean. Furthermore, instead of linearizing the system model, the input of the filter can be sampled around the mean. These samples, called sigma points, are propagated through the nonlinear system model. After this propagation, samples are recombined for the calculation of the new mean and sigma. This expansion is called the Unscented Kalman Filter [1].

The Extended Kalman Filter has a computational advantage over the Unscented Kalman Filter, but its complexity makes it unsatisfactory for real time applications in large areas. This complexity results from storing and updating all the correlations between the landmarks and the robot pose, which is explained in detail by Leonard [5]. This requirement is satisfied by updating all landmark correlations using all available information. Further improvements have been

made to the Extended Kalman Filter to reduce the computational cost involved. For this purpose, the Compressed Extended Kalman Filter was proposed by Guivant and Nebot [6].This filter updates the correlations in the current local area. After series of iterations, this information is propagated through the entire landmark set. The theoretical background of this algorithm is given in Section 3.2.3.

With each improvement to the Kalman filter, the algorithm provides a better performance under systems with different characteristics. For instance, the Unscented Kalman Filter provides a more precise solution than the Extended Kalman Filter despite the slightly higher computational cost. The complexity and performance analysis of the Extended Kalman Filter, Unscented Kalman Filter and Compressed Extended Kalman Filter under different computational powers were explained by Tuna [22].

The linearization problem can be solved using the Extended Kalman Filter or Unscented Kalman Filter, and the computational complexity can be reduced using the Compressed Extended Kalman Filter. However, there is still a detergency problem due to the errors related to the accumulated linearization. Julier [23] overcame these problems by not relying on the correlation information, and developing the Covariance Intersection method. This method is used to combine two estimates, which are consistent and optimal under unknown correlation cases [24]. The Covariance Intersection is more general and has a lower computational cost, but it does not provide the correlation information even if it is known. Therefore, a hybrid approach, Split Covariance Intersection, was proposed, which splits the information in the correlated and uncorrelated parts [25]. In this approach, the system is divided into two parts; known correlation and unknown correlation. This way, the state can be more precisely updated than the Covariance Intersection algorithm. In cases where the correlation information is fully known, the Split Covariance Intersection algorithm takes the place of the Kalman Filter approach. A detailed explanation of the Covariance Intersection algorithm is given in Section 3.4.3.

2.2.1. Feature Extraction

Feature extraction is the preliminary process of data association in feature-based SLAM applications. Sensor data is processed to define a meaningful feature structure. In this process, the selection of the sensor to be used is very critical since every sensor type has a different output measurement. For instance, cameras give an image of the captured scene, while SONAR only gives the distance measurement, which is used in Ruiz's work [26], and LIDAR gives the raw data of the distance measurement of the scanned angle range. According to the sensor output, the algorithm used for feature extraction also changes. For the camera output image, edge detection is the most popular feature extraction method, which can be performed using a Canny detector [27], Sobel or Prewitt operators. Moreover, there are also corner, blob, Scale Invariant Feature Transform [28] and Speeded Up Robust Features, used as detectors in the feature extraction process of images. LIDAR is widely used in indoor environments for grid-based SLAM applications since it is easy to relate the raw data obtained from the LIDAR with the map grids without feature extraction. In addition to being commonly used in indoor environments, LIDAR is also very useful in outdoor environments due to the high resolution and high distance accuracies. The feature extraction process for the LIDAR output is the same as those for image processing techniques, such as corner, line and tree detections. General purpose features were given by Li [29].

The main criterion for the selection of a sensor in SLAM is the environment. For instance, a camera, SONAR or low range LIDAR can be used for an indoor environment. On the other hand, LIDAR with its high resolution and range specifications can be more appropriate for underwater applications in the outdoor environment. The feature extraction method is then determined according to the selected sensor and the environment. In the outdoor environments such as parks, LIDAR with a tree detection algorithm is suitable while for indoor environments such as corridors, edge and corner detection algorithms are more appropriate.

2.2.2. Data Association

Data association is one of the critical parts of feature-based SLAM techniques. In this process, the information on the relation between the measurements and the landmarks is extracted and the matching measurement for the existing landmark is determined. New landmarks or false positives (outliers) can cause irrelevant measurements; therefore, the relation between a landmark and measurement should be carefully analyzed. An incorrect data association can lead to the algorithm initializing a wrong landmark. The information updated with the incorrect landmark measurement causes the robot to correct its state in a wrong manner. As result of these cumulative estimation faults, the algorithm diverges as explained by Neira [30].

A data association algorithm can be examined in two parts; a test part, which is used to determine the compatibility between measurements and features, and a selection criterion, which is used to choose the best compatible pairs. Therefore, the data association issue can be considered as a search problem in the space of measurement and feature correspondences.

In literature on SLAM, the Nearest Neighbor is a very popular algorithm used for data association due to its simplicity. This algorithm uses the distance between the landmarks and the measurement to determine the compatible pairs, and searches for the highest compatibility (shortest distance), which is the nearest neighbor. In this algorithm, measurements are individually tested for compatibility, so this algorithm is also called the Individual Compatibility Nearest Neighbor. The most popular compatibility test is based on Mahalanobis distance [31], which is also known as the normalized innovation squared [32]. The details of this algorithm and the Mahalanobis distance calculations are given in Section 3.1.1.

The complexity of the Nearest Neighbor algorithm is $O(mn)$, m being the measurement and n being the feature number. However, in this algorithm, the correlation between the measurements of the same vehicle results in an inconsistency in the compatible pairs. For instance, in loop closing scenarios,

robots explore a previously explored area, causing this algorithm to mismatch the measurements. Due to the accumulated estimation errors in the robot's position, the measurements calculated according to the robot pose result in displacement errors. All the measurements are affected by the error in the robot's position, which makes this algorithm inefficient for loop closing events. Despite this drawback, this method has the lowest complexity with respect to its competitors.

Under complex situations, such as spurious measurement and loop closing scenarios, the overall compatibility gains more importance for data association. The Joint Compatibility Branch and Bound algorithm was developed by Neira and Tardos [30] to reduce this complexity. They used an interpretation tree in their work, which also increased the overall consistency by taking into consideration the joint compatibility. The levels in the tree define a possible association with the measurement and the path from the root to the leaf of the tree indicates the possible compatible pairs, which gives the joint compatibility. The branch and bound algorithm searches the tree using the depth-first-search algorithm, and measures the maximum joint compatibility of set of pairings using the Mahalanobis distance. The bounding part of this algorithm means that if the compatibility fails in a certain node, its child nodes are not searched. A detailed explanation of the joint compatibility test calculations is given in Section 3.1.2.

The difference between the joint compatibility and individual compatibility results in different compatible pairs under the same scenarios. Figure 4 presents an example scenario involving a spurious measurement to clarify the difference between the behaviors of the joint and individual compatibility methods. Here, the stars represent the landmarks, and due to the static nature of landmark, z_2^3 is the spurious measurement.

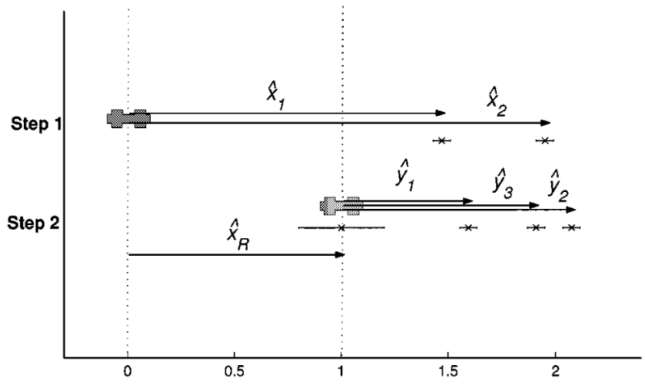


Figure 4 Mono Dimensional Spurious Measurement Scenario [30]

Figure 5 shows the different behaviors of algorithms under the scenario given in Figure 4. $\hat{x}_R$ shows the difference in the robot's position, and $\hat{x}_i, \hat{y}_i$ give the measurements. In the measurement notations, i represents the number of measurements. Each circle indicates the matching hypothesis with a range of uncertainty, the big square represents the uncertainty without correlations and the ellipsoid shows the uncertainty with correlations. The Nearest Neighbor algorithm chooses the $(y_1, x_1), (y_3, x_2)$ hypothesis since its center is the nearest to the center of the square. On the other hand, the Branch and Bound algorithm in the Joint Compatibility chooses the $(y_1, x_1), (y_2, x_2)$ hypothesis since it is the only hypothesis overlapping with the ellipsoid.

Despite the performance improvements brought by the Joint Compatibility, such as the implementation of the interpretation tree and the use of the branch and bound algorithm, the complexity problem prevails. For example, in environments with high frequency rated sensors, the complexity of algorithms is too high for real time applications. Therefore, the Nearest Neighbor remains to be a very popular data association approach for SLAM.

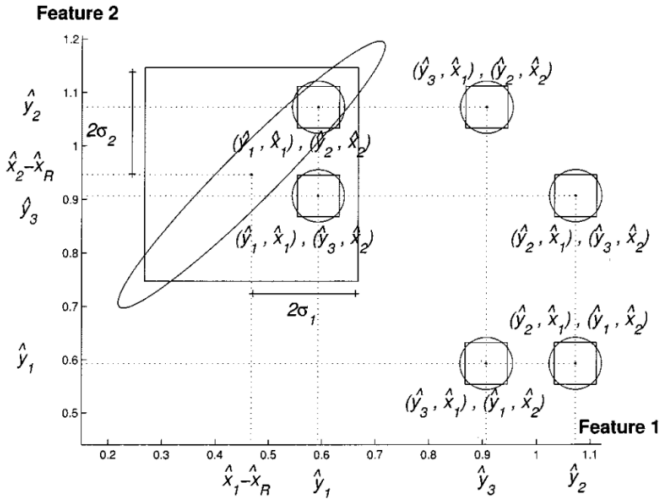


Figure 5 Illustration of Different Algorithm Behaviors under the Same Scenario [30]

2.3. Multi-Robot SLAM

The single-robot SLAM has been investigated in detail in recent years, and its robustness has been verified using the modified versions of probabilistic SLAM methods. However, there are still concerns regarding the time requirements and fault tolerance. For instance, if the exploration area is larger, the time required for mapping increases. Moreover, when working in more complex environments, a possible robot or algorithm failure results in the failure of the whole mission. On the other hand, multi-robot solutions have great advantages over single-robot solutions. For instance, with the increase in the number of robots, the time requirement reduces, and the fault tolerance and the accuracy of the resulting map increases.

The time advantage of multi-robot SLAM over single robot SLAM was examined by Marjovi [33]. In his paper, Marjovi used different number of robots in different environments, and obtained satisfactory results. For example, the mapping time of two robots was reported to be reduced by half when compared to the single robot mapping time. Despite these advantages, the multi-robot SLAM still requires the exploration of multiple issues, such as online path planning, simultaneous

localization and mapping, feature extraction and data associations. However, all have been examined in studies using the single-robot SLAM. In addition, task assignment, communication topology and map-merging issues are challenging issues for the multi-robot SLAM scenarios, which are explained in Section 1.1.

The task assignment and communication topology issues were not included in the scope of this study. However, map-merging was investigated using the unknown initial correspondence and limited communication assumptions. These assumptions make it easier to address the multi-robot SLAM problem as an extended version of the single case. Similarly, Thrun [1] examined the multi-robot SLAM using these assumptions. In his work, robots updated their maps with the information obtained from the maps of other robots by summing the information vector and matrix. Although, the initial position is known in this assumption method, there is still the problem of finding the matching (same) landmarks in two maps. This requires a proper data association algorithm. This problem also exists in scenarios, without the known initial position assumption, and data association problem needs to be examined as explained in Section 2.2.2. After matching the landmark set, the information needs to be merged. The reason for the merging process is to obtain the precise location of the landmarks. A detailed explanation of the map merging process is given in Section 2.3.2.

The unknown relative position assumption, on the other hand, makes it difficult to integrate the data since the maps cannot be directly merged due to the different frames of robots. Therefore, the first requirement to initialize the map merging process is the alignment of the maps using the global map transformation. A detailed explanation of this preliminary process is given in Section 2.3.1.

In the multi robot SLAM, different data fusion architectures can be used for the system design. To fully understand the map-merging problem in the multi-robot SLAM, first different types of data fusion architectures need to be examined. Three types have been defined with respect to their data fusion processing unit; centralized, decentralized and distributed.

Centralized systems have a central unit to process the all transferred data. This unit is the only unit in the system, which is capable of processing data. Other units are only responsible for collecting and transferring the raw data (sensor output). Therefore, this system architecture needs a longer communication range and more bandwidth. In addition, these network assumptions are not easy to satisfy in real world applications. Wireless channels are sensitive to failures, their communication range is limited and their limited bandwidth capacity is not sufficient to transfer a large amount of data. Moreover, the time delays in the data transfer of each robot cause critical problems in this type of architecture.

In contrast to the centralized systems, decentralized systems do not have central unit nor strong network assumptions. The main characteristics of decentralized systems given by Whyte and Stevens [34] are as follows:

- There is no need for a central fusion center for the operation to be successful. All nodes are individually responsible for data fusion.
- There is no common communication facility to broadcast information. All nodes are individually responsible for own one–to-one communication.
- There is no global knowledge of the network topology. All nodes have information about the neighborhood.

According to this description, nodes can be considered as robots in the decentralized systems. All the requirements for decentralized systems result in the scalability of the computational cost and the communication bandwidth. Moreover, as the number of nodes increases, the system becomes more robust against possible failures, and changes in the network structure [35]. Node failures are the faults in the algorithm and the robot breakdown issues. The network structure changes, such as the disconnection of one-to-one links, are the result of the changes in the positions of the robots and the communication range limitations. Although decentralized systems require a lower communication bandwidth than the centralized systems, there is still a problem due the information transfer between the robots, which is called as Fisher and Shannon's

measurements [36]. Moreover, an increase in the number of robots results in the loss of communication, which can be calculated by $O(n^2)$ for the each communication step. Here, nis the number of robots in the system.

In the distributed systems, all robots are responsible for processing the data. Therefore, instead of the unprocessed raw data, the information about the states of the robots are transferred to the fusion unit. The fusion unit is responsible for fusing the local states of robots into a global consistent state, which is the global map of the environment. In this architecture, multiple fusion units can be used. With this property, the distributed system designs can be combined with the decentralized architecture, which results in a hierarchical architecture.

All the mentioned architectures have advantages over one another. For instance, in theory, with the correct data association process and tolerable data transfer size, the centralized architecture is optimal. In realistic scenarios, however, decentralized systems have the advantage of allowing robot-to-robot communication, which results in a lower bandwidth requirement. However, data transfer requires a respectively higher bandwidth than the distributed design. In the distributed design, the states of the robots as well as the associative probabilities are used for fusion and they are more complex representations than the measurements. On the other hand, the states also contain the past information and this complex representation makes it more difficult to associate the current data with the previous data. In the decentralized systems, separating the old measurement from the new measurement is easier. The task sharing property of the distributed systems is another advantage. A detailed comparison between the decentralized and distributed systems using the multi-robot SLAM was given by Leung [37]. In his study, Leung used the centralized equivalent to compare the results of the two implementations. He found that the time requirement of the non-distributed implementation for the centralized equivalent estimate is lower than that of the distributed implementation. On the other hand, Leung noted that the distributed implementation decreased the computational load of the robots.

The decentralized and distributed architecture prevent the system from collapsing in complex environments; however, it is more costly due to the complexity of the initial design. The centralized system involves the transfer of a huge amount of data to update the whole system, and the computational cost of processing the whole sensor measurement of the central node is high [38]. The data transfer problem also occurs in the decentralized systems at a low level due to one-to-one nature of the information flow. Therefore, some improvements have been made to the centralized systems to overcome the problem of the limited communication bandwidth, such as transferring the most informative features of the map, as performed by Thrun [39]. In his work, Thrun extracted the most informative features by subtracting the previously transferred information from the current information on the state of the robot. Using the subtraction method, the most informative part of the map is extracted and limited by a constant value, which is the size of the sub-map. Similarly, in another study, Carlone [40] transferred the sensor measurements to the other robots. Obtaining this data, the robot augments its local map with the regions explored by other robots. However, the cost of this transfer is too high for limited bandwidth capacities. Therefore, the robots reset the history of the sensor measurements at the end of the transfer. On the other hand, in Cunningham's study [41], only the information about the landmarks, not the trajectory, was transferred between the robots. The longer trajectories of the robot can be a problem in the multi-robot SLAM due to the increase in the trajectory with the increase in time. Transferring only the landmark information prevents the divergence of the transfer load system. This shows that information selection is a requirement even for the decentralized systems.

To summarize, in practice there is no single architecture that is optimal for all different systems. The selection of the architecture should be performed according to the system requirements, such as the communication capacity and the processing ability of the robots.

In the distributed and decentralized systems, the map-merging procedure begins after the robots share their local information regarding the relative sensor

measurements or local maps with their neighbor robots within the communication range. In this thesis, this procedure will be examined in two main sections; global map transformation and the map-merging of the overlapped map.

Global map transformation is the first process when the initial positions of the robots are unknown. The unknown initial position assumption results in each robot creating a different local map frame since based on the assumption of its initial position. Therefore, a map alignment using the global map transformation is necessary. This can be performed using two methods. The first method is to calculate the relative positions of the robots using the sensor measurements from other robots, and then finding the overlapping areas using data association methods. An alternative is to transfer the local map information to another robot with the assumption that there is an overlap between their maps. After sharing this information, the robot searches the shared map to find the overlap using possible translation matrices. After finding the overlapping region, the true translation matrix is also found. These different approaches to the unknown initial problem of the multi robot SLAM are examined in Section 2.3.1.

After finding the relative positions of the robots, if there are overlapping regions on the local maps of the robots, the landmarks in these regions can be merged through the map merging process. In this process, the matched pairs in the overlapping regions are merged into one feature using their mean and variance estimations. Therefore, the positional information is more precise and accurate than the one that was previously obtained. In the literature, different map-merging techniques are investigated, which will be examined in Section 2.3.2.

2.3.1. Global Transformation

In the unknown positions scenario, the relative frames of robots need to be estimated. Global transformation is the process of finding the global transformation matrix. In this process, there are two main approaches. In the first approach, the robots estimate the position of another robot by measuring its relative position, and then find the transformation matrix from the relative frame.

In contrast, the second approach only uses the map of the other robot with the assumption that there is an overlap between their maps. In the first approach, after finding the transformation matrix, the landmarks can be transformed into the frame of the other robot. However, there is still a need to perform a data association, called the multi-robot data association, for which single-robot association techniques, such as Nearest Neighbor or Joint Compatibility Branch and Bound, can be used. The details of these association algorithms are given in Section 3.1. In the second approach, the data association process requires the calculation of the global transformation matrix. Therefore, search algorithms with map similarity heuristics are needed.

In both approaches, the transformation matrix and the matched landmark pairs are the outputs of the process. In this thesis, these approaches are examined in two sections; global transformation using the relative measurements and global transformation using the map overlap.

2.3.1.1. Global Transformation Using the Relative Measurements

In this approach, the robots meet at a pre-defined or random point on the map. Upon meeting, each robot calculates its distance and angle in relation to the other robot. These measurements are then transferred to each other for the calculation of the relative transformation, which is given in detail in Section 3.3.1. A camera is one of the most popular sensors used to find the bearing measurement of the other robot. For instance, in Kim's work [13], a checkerboard pattern on one robot is detected by the camera of another robot. Similarly, in Zhou's work [42], an omni-directional camera is used to detect the upper edge of a cylinder placed on another robot. The color, height and radius of the cylinder are known by all the robots. Using this priori information and the measurements, the bearing measurement to the target robot is calculated. Moreover, both robots shave a range sensor to obtain the range measurement of the other robot. By calculating the weighted average of the range measurements of each robot, a more accurate distance measurement is obtained. After finding the relative transformation between the robots, the global frame transformation is calculated using the estimates of the

states of the robots. Using this transformation, the map of each robot is moved to the global frame of the other.

Carlone's work [40] is another example of using sensor measurements to find the transformation matrix between robots. In his work, Carlone used a pan-tilt camera for angular measurement and a laser range sensor to obtain the relative distance from the target robot. The same formulation is used in Zhou's work [42] for the calculation of the global frame transformation. After finding the global transformation matrix, not the maps but the history of sensor and odometer measurements is transferred to the other robot. Using this information, the robot processes the odometer measurements as if it was traveling backward on the trajectory of the other robot. In this process, the sensor measurements related with the trajectory are also used to extend the map. In this scenario, robots only transfer their own sensor measurements due to bandwidth limitations. Moreover, the previous meeting (rendezvous) time of each robot is used to transfer only the recent difference in measurements. This not only prevents the system from double counting the information, but also uses less bandwidth. In this scenario, data association is not required, since a feature-based map is not used.

After finding the global transformation matrix, which is explained in detail in Section 3.3.1, multi-robot data association can be simply performed using any of the single-robot data association techniques. When maps are transferred to the other robots, the landmarks can be used as sensor measurements in the single robot data association. For instance, in Zhou's work [42], landmark pairs are found by using the Nearest Neighbor algorithm, which is explained in detail in Section 3.1.1.

2.3.1.2. Global Transformation Using the Map Overlap

In this approach, robots transfer their map information with the assumption that there is an overlap between their maps. The main purpose of this approach is to find the best transformation by selecting the best overlap from the transferred information. Figure 6 illustrates the overlapping regions of two maps m_1 and m_2

in two different transformations. Here, the second illustration shows a perfect match between the maps.

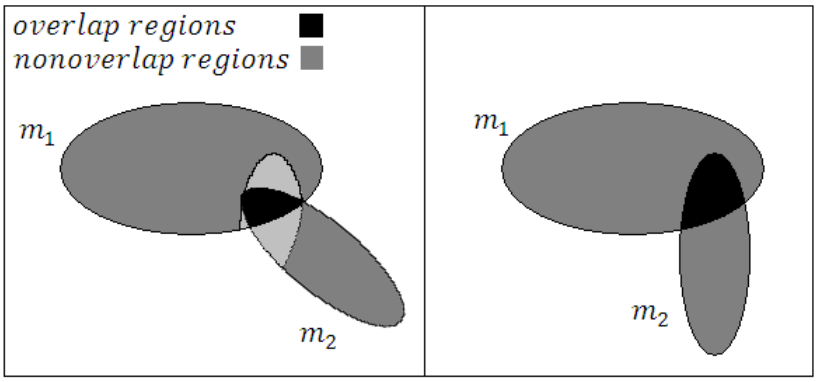


Figure 6 Illustration of the Effect of Transformation on the Overlapping Area

The search for the best overlap in a robot's feature set can also be referred to as the multi-robot data association problem. This association problem is similar to the single-robot data association problem when the initial positions of robots are available. Instead of using features extracted from the sensor measurements, the feature sets of other robots can be used, which is explained in detail in Section 2.3.1.1. However, the lack of prior knowledge about the position of other robots causes the search for the best association to be performed under different transformations, which makes the association more difficult than it is for the single robot case. Therefore, the possible transformation space is also considered in the multi-robot data association problem.

The search space of possible transformation is too large for even moderate-size maps in real time applications. Therefore, different preprocessing techniques have been developed to reduce this space. These techniques use the structure similarity between the maps for the elimination of possible transformations. A global transformation including this preprocessing can be examined under two sections; map similarity and multi-robot data association.

In contrast to the relative measurements approach, finding the global transformation matrix through the overlapping points does not require a rendezvous between robots. In this approach, being within the communication range is enough to share the map information. Therefore, this approach has a better chance of a successful map-merging in complex environments. For instance, in the indoor environments such as labyrinths, buildings and underground tunnels, sensor limitations due to features, particularly walls, prevent robot-to-robot measurements. Under these circumstances, using communication for the exchange of map information is more suitable than using the relative sensor measurements.

2.3.1.2.1. Map Structure Similarity

Map structure similarity is the process of finding similar parts between the robots. These similar parts are possible overlapping regions between the maps. This similarity can be found using the map structure or geometric information. A map structure contains the characteristics of features, and geometric information is the relation between these features, which are meaningful in the feature-based map SLAM application. However, it can also be performed in grid-based maps with the use of feature extraction algorithms. The characteristics of the features on a map include color, shape and length. The edge lengths between features are a simple example of geometric information that can be extracted from the maps. These properties are used to reduce the search space of the global transformation.

In the literature, different techniques have been reported to find the structural similarities between the maps. For instance, Dedeoğlu and Sukhatme used the extracted features to find the overlapping areas [43]. They classified the landmarks into two main types; node and link. In their study, corners, junctions and doors are classified as node while open spaces and blocks are classified as link-type landmarks. The search space is reduced by using feature properties with two heuristic functions; pairing up only the same type attributes and considering only the rare features of candidates. However, in their study, feature classification is done manually and their probability distributions are not considered.

In Konolige's work [44], a similar technique was used with some improvements. Features, such as corners, doors and junctions, and scan patches consisting of a recent set of scans were used to determine the similar regions between the maps. The similarity was scored by using the likelihood metric, which is calculated from the likelihood and feature likelihood multiplication of the scan readings. The means of the uncertainty and errors between the maps was used for the likelihood estimation of the scan readings, and the distance and angle in relation to the nearest feature were used to estimate the feature likelihood. Using the structure of the map considerably improved the performance of the transformation search. However, feature extraction was not examined and performed manually.

To summarize, all the above-mentioned methods for map sharing focus on the importance of the map structure in finding the step where the overlap occurs. However, all methods require different feature sets to be compared in this step. These features were manually labeled in Konolige's work [44], and improved by Dedeoğlu and Sukhatme using compass readings [43]. With these features, algorithms are enhanced for a better performance. However, in general scenarios, extraction of these features is not possible. Therefore, it is more suitable to use the geometric information from the map to find the overlapping regions not to lose the generality.

In Huang and Beevers's work [10], the feature extraction issue was addressed by using a topological map structure. In topological maps, additional information on the degree of vertices and orientation of edges at vertices are stored in a graphical structure. The vertexes are the nodes representing the positions of the robot and features, and edges are the paths connecting one vertex with another using the measurements. This structure consists of exact features, such as the degree of vertex, and inexact attributes, such as the length of the edge. In the calculation of the map similarity, exact features are compared to find a perfect match, but inexact features are compared using the similarity test. The similarity test is performed by assuming the Gaussian error for the angular orientation and distance. Using these matching tests, the algorithm decides whether it will accept

the candidate vertex pair and angle as a hypothesis. After the hypothesis generation step, the hypothesis consistent set is searched by extending the vertex to sub-graphs. These sub-graphs are the representation of possible overlapping regions. In fact, Huang and Beevers's study shows that using only the structure of the map does not give satisfactory results for self-similar environments, so using the geometric information improves the merging performance.

Saeedi [45] examined the grid-based map representation for the map-merging problem. However, he suggested that without any preprocessing step, merging maps with higher dimensions such as grid-based maps is not suitable for real time implementations. In his work, Saeedi transformed the grid representation into a reduced topological map representation using the self-organizing maps method. This method is used to extract meaningful information from a map to accelerate the map merging process. After this process, the map representation consists of cluster points and cluster surfaces. Clusters points are the meaningful extracted features and cluster surfaces are the links between these features. These extracted features help find the norm vectors of surfaces. Birk and Carpin [11], who did not use the map similarity preprocessing, obtained similar results and reported the time requirement to be ten times less .

The contribution of the preprocessing step using the meaningfully extracted features was given by Saeedi [45]. However, these algorithms still need too much time to finish the map-merging process due to the segmentation and clustering requirements. In contrast to the grid-based map representation, the feature-based map representation has a complex map structure. This complex nature of the feature-based map was examined by Cunningham [41]. In his work, Cunningham used the Delaunay Triangulation and a simple similarity metric to find the map similarity, which is explained in detail in Sections 3.3.2.1 and 3.3.2.1.2. These geometric features are extracted from the map by obtaining a unique triangle set using the Delaunay Triangulation algorithm. The circumference and area found by this algorithm is then used to calculate the similarity metric. This metric reduces the search space of the transformation matrix.

To summarize, all the mentioned algorithms are used to improve the search performance, and the geometric information is important for the map structure. For example, in the maps with similar structural properties, such as doors, windows and corners, the map similarity cannot reduce the search space. However, using the geometric information gives better results since it gives information on the relationship between the features. However, if the environment has similar geometric patterns, even geometric information may not improve the transformation search performance. In such maps, using outlier-tolerant search algorithms provides a better performance due to the similar structural and geometric patterns on the map in false transformations.

2.3.1.2.2. Multi-Robot Data Association

In this section, different multi-robot data association algorithms that have been reported in the literature are discussed. One of the maps is kept stable while the other map is transformed using the possible transformation matrix. Following the transformation, the performance of the transformation matrix is calculated using the performance of map overlapping. In the literature, different techniques have been reported for the evaluation of this performance; such as the similarity of the overlapping regions and the count of compatible features. For the calculation of candidate transformation, one candidate pair is selected. If this pair has the heading information, this means that it can provide sufficient data for the calculation of transformation in the closed form. However, if only position information is available, then two candidate pairs are required. This calculation is explained in detail in Section 3.3.2.2.1. After finding the candidate transformation matrix, the best transformation between the two maps is explored using different search algorithms. In the following parts of this section, transformation search algorithms are discussed in terms of their performance metrics in evaluating the transformation.

In Dedeoğlu and Sukhatme's work [43], landmarks have heading and position information, so each pair has enough information for the calculation of the candidate transformation between the local frames of the robots. After finding the

candidate transformation for all possible landmark pairs, this transformation is applied to the rest of the landmarks in the data set. The matched landmark pairs are counted and the transformation with the highest number is selected. Dedeoğlu and Sukhatname did not use any search criteria; instead, they scanned the whole search space. Moreover, in their study, the probabilistic distributions of the landmarks are not considered, so the Euclidian distance is verified using the threshold value for the compatibility of the possible landmark pairs.

In Birk and Carpin's work [11],transformation is guided with Carpin's Adaptive Random Walk algorithm [46]. In this algorithm, randomness is obtained using the Gaussian distributed random selector and the adaptive part is obtained through image similarity and overlapping function. Image similarity is calculated using the distance map between two maps, which is an array of the Manhattan-distances to the nearest point with values in the map. The overlapping function is the measurement of the agreement between the two maps. The Adaptive Random Walk algorithm searches the configuration space and updates its transformation matrix with its heuristic function or a randomly generated sample set. In Birk and Carpin's heuristic functions, the overlapping area and map similarity are used to associate the overlapping regions in grid-based maps. This search algorithm is explained in detailed in Section 3.3.2.2.3. Birk and Carpin's main motivation behind using the random walk algorithm for the search of the possible transformation space is to prevent the algorithm from being stuck in a local maximum. Moreover, the adaptive part of this algorithm accelerates the search for a better and larger overlap by using the heuristic function value. A detailed explanation of this search algorithm is given in Section 3.3.2.2.3.

In Saeedi's work [45], the grid representation is transformed into a reduced topological map representation. With the use of the features extracted from the topological map, norm vectors of surfaces and points are found. The directions of the norm vectors of the cluster surface are put into a 360-degreehistogram, and then the relative rotation part of the global transformation is determined using this histogram. In the calculation of the relative translation part of the global

transformation, the norm vectors of the cluster points are used by searching a possible transformation. Saeedi implements a search algorithm similar to the Iterative Closest Point algorithm, which iteratively updates the transformation matrix with the correspondence set in that iteration. The details of the algorithm are given in Section 3.3.2.2.4.

In Cunningham's work [41], features with their probability distribution information are used to detect the overlaps. Similar Delaunay Triangle center points and the Random Sample Consensus (RANSAC) algorithm are used to search the transformation matrix between the maps. This algorithm calculates the candidate transformation matrix from the similar triangle center pairs and checks the search space for a match between the landmark pairs. This search space is the output of the map structure similarity, which gives similar triangle center points. The matched landmark pairs are obtained using a compatibility test (Mahalanobis distance) under candidate transformation, and their count is used for the evaluation of the performance of the transformation matrix. A detailed explanation of this compatibility test is given in Section 3.1.1, and the RANSAC algorithm flow chart is explained in Section 3.3.2.2.2.

To summarize, the multi-robot data association issue is different from the single-robot data association issue. For instance, the compatibility test in the multi-robot association requires the use of map similarity metrics, such as the correspondence set size, overlapping area size or Cunningham's similarity metric. The reason behind this requirement is that the transformation between maps is unknown. Therefore, map similarity is used link the maps together. Moreover, search algorithms, such as the Adaptive Random Walk, Iterative Translation Search or RANSAC, have adaptive or random characteristics due to the possibilities of different overlapping regions. These possibilities are randomly distributed due to the random initial positions of robots. Therefore, the multi-robot data association is a more difficult process.

2.3.2. Map-Merging

In this section, map-merging algorithms proposed in the literature are examined in terms of their use in different map representations. The map-merging process in a grid-based map representation is obtained by summing the probabilities of the matched grids. The matched grids are found using the same occupied index of the map. The grid probabilities are summed if they have the same properties in terms of being occupied or free. If one of the grids in a matched pairs conflicts with the other, these grids are marked as unexplored. The information of a pair being occupied or free is kept in case there is no matching pair. The illustration of the map-merging process of two grid maps is given in Figure 7.

In contrast to grid-based maps, the merging process for the feature-based maps needs to combine the distribution functions of the matched landmarks, which is the output of the global transformation process. The matched landmarks are the same landmarks that exist in both maps. They need to be calculated using a data association algorithm, which associates the landmarks in one map with the landmarks in the other map. The mean and covariance matrices of the landmarks are used for the fusion of these landmarks in the probabilistic map-merging process. Through this fusion, more precise and accurate positions can be obtained. Figure 8 shows the change in the covariance matrices after map-merging.

The map-merging process in feature-based maps has similar characteristics to the sensor data-fusion process. In sensor data-fusion process, every sensor data is combined and the precise location of each feature is obtained. The following sensor data fusion approaches have been reported in the literature;

- Extended Kalman Filter,
- Maximum Likelihood Estimator,
- Modified M-Estimator,
- Covariance Intersection Estimator,
- Orthogonal Gnanadesikan-Kettenring Estimator,
- Hybrid Covariance Intersection and Orthogonal Gnanadesikan-Kettenring,

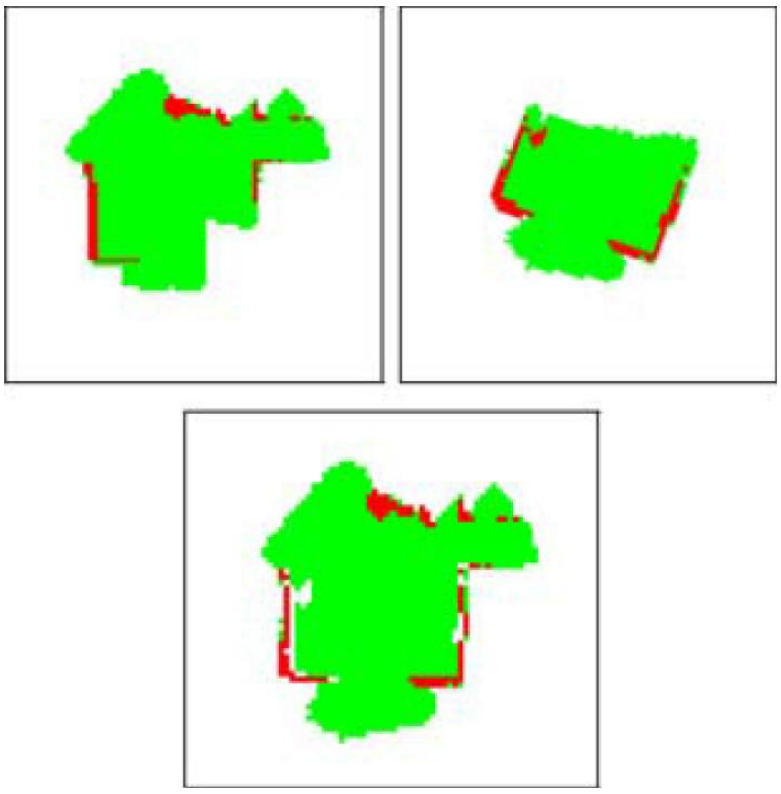


Figure 7 The upper two maps explored by different robots are merged into the lower resultant map [11]**.**

Centralized, decentralized and distributed systems have different characteristics, which are explained in detail in Section 2.3. Therefore, the above-mentioned solutions to the map-merging issue need to be investigated in relation to the characteristics of different systems.

In centralized systems, the information fusion in the Extended Kalman Filter solves the map-merging problem using the initial knowledge of the dynamics of all robots. However, in this solution, the correlation values between the maps are updated by a central unit. In addition, the memory and time requirement makes this approach not suitable for real time implementations and the system behaves similarly to the single-robot SLAM. Therefore, in this thesis, the Kalman Filter was not used to perform the map-merging process.

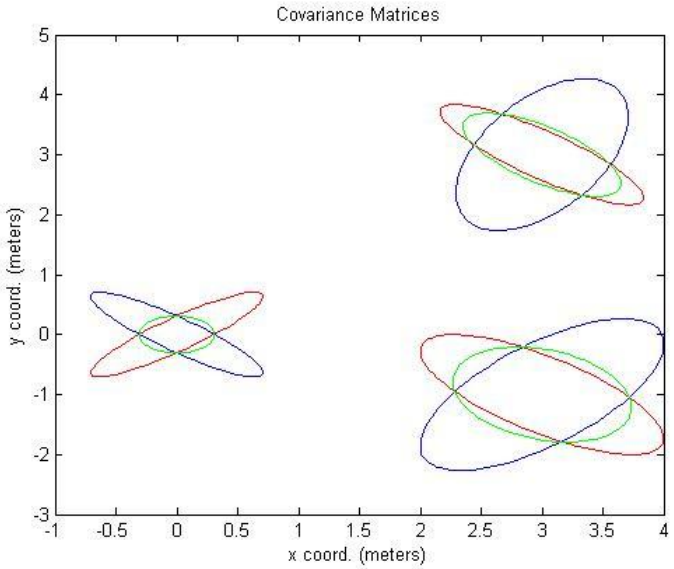


Figure 8 Ellipses with a larger area are the landmark covariance matrices in different maps, which are merged with the smaller (inner) ellipses.

In decentralized systems, there is no central unit or a fully connected network. Therefore, a more detailed (complex) design is required for map-merging. For instance, using the Kalman Filter requires an independence assumption between the nodes. However, this assumption can result in divergence for two main reasons; unpredictable correlated system noises and the correlation resulting from previous information flow between the robots. The first can be prevented by modeling the system more precisely. However, the solution to the second case requires maintaining all the correlations between the nodes. For instance, channel filters [34] can be used to maintain the correlations, but these filters cannot work with cyclic network connections, in which the information can flow in multiple directions. Therefore, using the Kalman Filter in decentralized systems is not appropriate. A detailed examination of the use of Kalman Filter in decentralized systems can be found in Julier and Uhlmann's work [47]. In their study, the use of Kalman Filter with channel filters in decentralized systems with a cyclic network connection was reported to be "impossible".

The specifications of a decentralized system require implementing the estimators without the correlation information or maintaining a global map. Maintaining a global map means that the robot does not integrate the information obtained from the other robots into its local map; instead, it clones its local map and merges both information in another map (the global map). Despite being costly, maintaining the global map, gives the opportunity to use different map-merging techniques.

In the map-merging process, if the covariance matrices of the features are ignored or the sensor outputs are stored for the data fusion process, the problem becomes an inverse probability problem. In inverse probability problem, main concern is to find the distribution of the data. For instance, if a coin is flipped five times, what is the probability of getting four tails? This problem is represented by $p(\text{Data}|\text{Model})$. On the other hand, in the inverse probability problem, the main concern is to find the model parameters for the observed data. Therefore, the representation of this problem becomes $p(\text{Model}|\text{Data})$.

The Maximum Likelihood Estimator is a very popular solution to the inverse probability problem. It is a conservative method; therefore, it assumes that the input data has no outliers and tries to minimize the errors without considering the outlier possibility. Moreover, an initial model is needed to find the parameters for this model. The distribution function of this model is minimized by taking the logarithm and the derivative. A detailed explanation of the Maximum Likelihood Estimator is given in Section 3.4.1.

M-Estimators are another popular class of estimators, which use the functions of data to obtain the minima. The estimation functions are the derivatives of the likelihood functions with respect to data. The M-estimator was first proposed by P.J. Huber [48] in 1981, who generalized the Maximum Likelihood Estimation [49]. There are also different types of robust estimators in the literature, such as the R-Estimator, L-Estimator and S-Estimator. These estimators and their theoretical derivations were explained in terms of their evolution Andersen's work [50]. These methods can be used in the decentralized systems by transferring the

sensor output history for data fusion. They can also be used with the extracted features without their covariance matrices. However, bandwidth limitations and the specifications of the distributed system make it difficult to investigate the M-estimators in detail. In addition, a solution involving these estimators causes information loss since it does not use the covariance information of the extracted features. Therefore, in the experiments in this study, the Maximum Likelihood Estimator was used as the base estimator.

In this thesis, in addition to the base estimator, a special type of M-Estimator, which assigns weights landmarks based on their covariance matrix, was used for map-merging. This technique uses the independence between variables to use additivity of statistical information, which is used for fusion of two sensor outputs in the lecture notes of Zisserman [51]. Then, in his study, Zisserman reported that weights cause large covariance matrices resulting in the uncertainty of the information regarding the landmark position. Using these weights, the resultant positions of the landmarks can be estimated. In this thesis, this algorithm will be referred to as the Modified M-Estimator. A detailed explanation about the implementation of this algorithm is given in Section 3.4.2.

Another estimation method similar to the Modified M-Estimator is the Covariance Intersection, which is more suitable than the Modified M-Estimator for distributed and decentralized systems. This algorithm does not use the correlation information between the features. In addition, its consistency has been proved for any degree of correlation in the study by Julier and Uhlmann [47]. In their work, Julier and Uhlmann also explained in detail, the advantages of the Covariance Intersection algorithm over the Kalman Filter in decentralized systems. This algorithm uses the convex combination of means and covariance matrices of random variables. In contrast to the Modified M-Estimator, it can also weigh the data to adjust the determinant or the trace of the resultant covariance matrix. A detailed explanation of this algorithm is given in Section 3.4.3.

Sequeira et al [52] used the Orthogonal Gnanadesikan-Kettenring Estimator to overcome the problem of unused correlation values in the Covariance Intersection algorithm. This method is based on the use of a robust estimator developed by Gnanadesikan and Kettenring [53], which uses actual measurements and their estimated covariance matrices, and modified by Sequeira et al [52]. Using these inputs, outliers are disregarded in the calculation of the final covariance matrix. A disadvantage of this algorithm is that it needs the whole data set to determine the outliers with their estimated covariance matrices. This process uses a high bandwidth; therefore, it is not suitable for distributed applications, but can be used in decentralized systems. The pseudo code of the algorithm and its explanation is given in Section 3.4.4.

In their study, Sequeira et al [52] also used a hybrid of the two estimators, the Orthogonal Gnanadesikan-Kettenring and Covariance Intersection to estimate the covariance.. They then compared their performance based on the area of the covariance matrix and chose the better covariance matrix as the final value. Their study also includes the performance results obtained from different data sets. These results show that in different data sets one estimator can outperform the other; therefore, a hybrid algorithm improves the performance in more general applications. Despite this improvement, a hybrid implementation involves calculating both covariance matrices with their performance metrics to make the final decision. This takes more time and increases the bandwidth and storage costs, which is not preferable in distributed systems. However, in decentralized systems using this hybrid algorithm improves the performance of the Covariance Intersection algorithm with the outlier rejection property of the Orthogonal Gnanadesikan-Kettenring estimator.

To summarize, different data fusion techniques have been developed and investigated for the map-merging process of the multi-robot SLAM. As mentioned above, all the algorithms have their advantages for different system designs. For instance, using the Kalman Filter update in centralized system architecture is the optimal solution. However, since this requires an extra process

in decentralized systems, robust estimators, such as Covariance Intersection or Orthogonal Gnanadesikan-Kettenring are more suitable than the Kalman Filter. On the other hand, in distributed systems, sensor outputs cannot be shared with other robots, so using simple and consistent methods, such as the Covariance Intersection algorithm is more appropriate.

2.4. Evaluation of the Map Performance

Maps generated by different SLAM algorithms represent the environment in different ways, as explained in Section 2.1. These differences in representation result from the purpose for which they are employed, which results in the need to use different methods for the evaluation of the maps. In the literature, there are different map quality criteria. These criteria are also called metrics and are not only used in academic studies for comparison of algorithms as in this thesis, but also used in competitions such as RoboCup Rescue [54], and Magic [55].

Despite the variety of metrics in the literature, only few can be used in general applications. Most studies on the evaluation of metrics use only one criterion, which is directly related with the application area. Therefore, other problematic areas are disregarded. In his study [56], Lee addressed this issue by introducing certain attributes to describe a metric for different mapping applications and suggested that a metric must;

- be clearly defined,
- be multi-valued,
- reflect the purpose of map,
- balance the coverage and detail, and
- be applicable during the construction of the map.

A clearly defined metric does not contain any subjective judgments. Having the multi-valued property means that the performance evaluation is not based on the "true/false" value. The reflection of purpose indicates that the metric contains what is important for the given application. Balancing the coverage and detail property is used to weigh the criteria in metric calculation. The balancing property

can be used between any of the properties. The last property of the metric being applicable in the construction of the map means that the robot can use the map when constructing it and evaluate its performance. All these properties give a clear description of a good metric for map evaluation.

Map evaluation algorithms are classified into three main classes by Schwertfeger [57]; path-based, place-based and structure-based. In path-based approaches, not the map information but the robot's path performance is used for evaluation. In such evaluations, the ground truth trajectory is required for comparison, which is hard to satisfy, particularly for outdoor environments. In place-based approach, the extracted map is compared with the ground truth data of the map, which is a more general evaluation technique. Structure-based approaches compare the resultant map structure with the ground truth map-structure, which needs a structural mapping algorithm, such as topological maps. All these approaches are explained in detail in Schwertfeger's work.

Map merging performance evaluation is similar to the map performance evaluation. Instead of using different SLAM, feature extraction or data association algorithms, the map-merging algorithms are compared using the merged maps. Therefore, the metric criteria and approaches for map evaluation can also be used for the evaluation of the map merging performance. However, path-based approaches are not applicable to this evaluation.

In this section, feature-based approaches in literature are discussed, and using Lee's metric properties, a metric that is appropriate for the application is selected to evaluate the performance of map-merging.

In Yairi's work [58], the Least Mean Squares of Euclidean Distances are used for map evaluation. Then, the Euclidean distance between the map features and the ground-truth is calculated. However, the covariance information is not included in the evaluation metric. Similarly, in Wagan's work [59], different feature extraction algorithms are compared using the feature-based approach. In his work, A pair

wise matching of features is performed based on the distances between the features. The count of the matched pairs is used as the metric for evaluation.

In Schwertfeger's work [57], artificial markers are placed in environment, and the resultant map is compared with the ground truth information of these markers in terms of accuracy. The accuracy of markers is calculated using the distance of markers to the ground truth positions of them, and the local consistency is calculated by using the distance between the marker pairs on the resultant map. These distances are subtracted and assigned to the consistency metric. Coverage is also used as a metric, which is calculated from the ratio of the count of the explored markers to the count of all markers. Combining these different metrics, the current study aims to satisfy Lee's metric property definitions and obtain a general metric definition. However, this evaluation metric only uses the mean values of the markers' position. This kind of evaluation does not consider the probabilistic distributions of features, which is very important information for SLAM.

On the other hand, the estimated covariance matrix is used for metric calculation in Klippenstein's work [60]. However, only the robot's position is used for error calculation due to the lack of true positions of the extracted features. In his study, Klippenstein used the Normalized Estimation Error Squared to calculate the errors, and the χ^2 acceptance test to check the consistency between the estimated errors and the covariance. After running the Monte Carlo algorithms on the filter, the average value is tested with a 95% confidence region of the χ^2 distribution. However, since this test of inconsistent values of feature extraction algorithms only gives true or false values, it is not considered a good evaluation metric. Therefore, the volumes of covariance ellipses are used as a metric for ranking. In the calculation of the metric, the volume of the covariance matrix is estimated using the robot pose. This technique involves the distribution of the estimated position of the robot, which is the most important input of the estimation of the next position in probabilistic SLAM algorithms. Therefore, in the current study,

this technique has been chosen for the metric calculation and to evaluate different methods for map-merging as explained in Section 5.

In this study, rather than the robot's position, features of the merged maps were used to compare the map-merging algorithms. The estimated feature and ground truth pairs were found using the Nearest Neighbor algorithm, as explained in Section 3.1.1. These pairs (ground truth and estimation) were then checked in terms of their consistency by using the Normalized Estimation Error Squared and the χ^2 acceptance test, which is explained in detail in Section 3.5.1. For the pairs that passed the test, the metric description of Klippenstein was used to evaluate the different map-merging algorithms for two-dimensional maps given in Section 3.5.2. An evaluation using this metric includes the covariance information, which is one of the main parameters in map-merging algorithms.

CHAPTER 3

THEORETICAL BACKGROUND

3.1. Data Association

In this section, Individual Compatibility Nearest Neighbor and Joint Compatibility Branch and Bound algorithms, which are introduced in Section 2.2.2 are examined in detail.

3.1.1. Individual Compatibility Nearest Neighbor

In this section, Individual Compatibility Nearest Neighbor data association algorithm is explained. Mahalanobis distance calculations, which are used in compatibility test, are also given in the following equations. The pseudo code of the algorithm is given in Table 1.

Table 1 Pseudo Code of Individual Compatibility Nearest Neighbor

Individual Compatibility Nearest Neighbor (P_N, P_M)
1: for $i = 1 \rightarrow n$
2: for $i = 1 \rightarrow m$
3: $d[i,j] = \text{distanceFunc}(p_i, p_j)$
4: for $i = 1 \rightarrow n$
5: $\text{dist}[i] \leftarrow \infty$
6: for $i = 1 \rightarrow m$
7: if $i \neq j$ and $d[i,j] < \text{dist}[i]$
8: $\text{dist}[i] \leftarrow d[i,j]$
9: $\text{NN}[i] \leftarrow j$

In Table 1, P_N and P_M are the position vectors of two set of features, which are the parameter of the algorithm, N is the number of first set, M is the number of second set. distanceFunc() is the compatibility test function, which is selected as Mahalanobis distance. NN[i] is the compatible set of features, which are selected as nearest neighbor. In this algorithm, distance values are calculated and stored, as mentioned in line 1, 2, 3. By using these values nearest pair is selected with the smallest distance value, as mentioned in line 7.

Mahalanobis distance function calculations are related with implicit measurement function, which is given in the equation (1).

$$f_{ij_i}(\bar{x}, \bar{y}) = 0 \tag{1}$$

In equation (1), $\bar{x}$ and $\bar{y}$ are the true values of feature and measurement, f_{ij_i} represents the relative location between feature i and corresponding measurement j_i. Accumulated errors in the position of features results this equation into the equation (2).

$$f_{ij_i}(x, y) \cong h_{ij_i} + H_{ij_i}(\bar{x} - x) + G_{ij_i}(\bar{y} - y) \tag{2}$$

In equation (2), h_{ij_i} is the innovation of pairing between measurement and feature, H_{ij_i} and G_{ij_i} are the first derivatives of the innovation matrix with respect to measurement and feature respectively. From these equations covariance matrix can be calculated as mentioned in the equation (3).

$$C_{ij_i} = H_{ij_i} \mathrm{Cov}(\bar{x} - x) {H_{ij_i}}^T + G_{ij_i} \mathrm{Cov}(\bar{y} - y) {G_{ij_i}}^T \tag{3}$$

By the use of resultant covariance matrix C_{ij_i}, following innovation test, which measures the Mahalanobis distance can be written as in equation (4).

$$D_{ij_i}^2 = h_{ij_i}^T C_{ij_i}^{-1} h_{ij_i} < \gamma_2 \tag{4}$$

In equation (4), $D^2_{ij_i}$ is the Mahalanobis distance between feature i and measurement j, and γ_2 is the compatibility gate, which accepts the 95% correct association for value 6.

3.1.2. Joint Compatibility Branch and Bound

In this section, Joint Compatibility Branch and Bound data association algorithm is explained. Joint compatibility test calculations, are given in the following equations.

The consistency of the hypothesis $H_i = \{j_1, \dots, j_i\}$, which is joint compatibility is obtained by using joint implicit function given in equation (5).

$$f_{H_i}(\bar{x}, \bar{y}) = 0 \tag{5}$$

In equation (5), $\bar{x}$ and $\bar{y}$ are the true values of feature and measurement, f_{H_i} represents the relative location between feature i and corresponding measurements in hypothesis H_i. Accumulated errors in the position of features results this equation into the equation (6).

$$f_{H_i}(x, y) \cong h_{H_i} + H_{H_i}(\bar{x} - x) + G_{H_i}(\bar{y} - y) \tag{6}$$

In equation (6), h_{H_i} is the innovation of pairing between measurement and feature, H_{H_i} and G_{H_i} are the first derivatives of the innovation matrix with respect to measurements and features respectively. From these equations covariance matrix can be calculated as mentioned in the equation (7).

$$C_{H_i} = H_{H_i} \mathrm{Cov}(\bar{x} - x) {H_{H_i}}^T + G_{H_i} \mathrm{Cov}(\bar{y} - y) {G_{H_i}}^T \tag{7}$$

By the use of resultant covariance matrix C_{H_i}, following innovation test, which measures the joint innovation test can be written as in equation (8).

$$D^2_{H_i} = h^T_{H_i} C^{-1}_{H_i} h_{H_i} < \gamma \tag{8}$$

In equation (8), $D^2_{H_i}$ is the joint innovation and γ is the compatibility gate, which accepts the 95% correct association for value 6.

3.2. Filters for SLAM Purpose

In this section, filters mentioned in Section 2.2 are explained in detail. Kalman Filter, Extended Kalman Filter and Compressed Extended Kalman Filter are investigated under feature based SLAM specifications.

3.2.1. Kalman Filter

Kalman Filter is used for estimate the state of system, which corrects the estimate by using feedback of measurements. This filter is proved optimal estimator under its assumptions. These assumptions are Markov assumption, independent zero mean Gaussian distributed noises, normal distributed priori state, and linear state and measurement models. These assumptions' details and Kalman filter optimality proof can be found in Thrun's work [1]. Kalman Filter has two main steps, such as time and measurement update steps, which are called as prediction and correction states. In Figure 9, the state transition diagram between these two steps is represented, which shows that time update transition can occurs independent of measurement update, also they can occur concurrently.

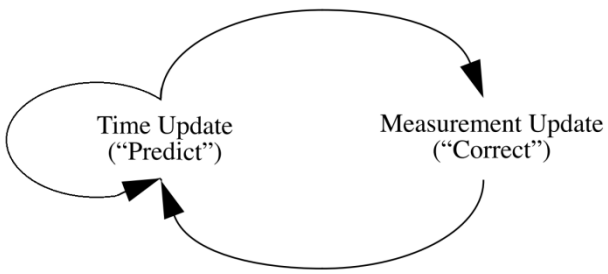


Figure 9 State Cycle Diagram of Kalman Filter

The estimated state consists of positions of robot and observed landmarks in the map, which is given in the following equation (9).

$$x_k = \begin{bmatrix} x_R \\ y_R \\ \emptyset_R \\ x_{L_1} \\ y_{L_1} \\ \vdots \\ x_{L_N} \\ y_{L_N} \end{bmatrix} \quad (9)$$

In equation (9), $x_R, y_R, \emptyset_R$ are the x coordinate, y coordinate and direction values of the robot pose, x_{L_i}, y_{L_i} are the x and y coordinate of the observed landmarks L_i, and N is the total number of landmarks observed.

Linear state time update equation is represented in formula (10).

$$x_k = Ax_{k-1} + Bu_{k-1} + \omega_{k-1} \quad (10)$$

A is (nxn) the state transition matrix, represents the transition from state x_{k-1} to x_k, B is control matrix (nx1), which relates the input u_{k-1} with state x_k, w_{k-1} is the independent Gaussian noise with $p(w) \sim N(0, Q)$ distribution.

Linear measurement update equation is represented in formula (11).

$$z_k = Hx_k + v_k \quad (11)$$

H is the observation matrix (nxn), which relates the measurement with state x_k, and v_k is the independent Gaussian noise with $p(v) \sim N(0, R)$ distribution.

The given model above result the following equations (12), (13) for predict state.

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \quad (12)$$

$$P_k^- = AP_{k-1}A^T + Q \quad (13)$$

A and B are the same in equation (10), P_{k-1} is posteriori estimate of state covariance matrix (nxn).

The given model above result the following equations (14), (15), (16) for correction state.

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} \quad (14)$$

$$\hat{x}_k = \hat{x}_k^- + K_k (z_k - H\hat{x}_k^-) \quad (15)$$

$$P_k = (I - K_k H) P_k^- \quad (16)$$

In these equations, K_k is the Kalman gain at step k, which is calculated with equation (14), for detail derivation of Kalman gain Thrun's work [1] can be read. Equation (15) is the exact correction step, which updates x_k with Kalman gain multiplied measurement prediction. The critical point at this step is measurement correction on x_k is weighted with the inverse covariance, which means if the measurement is more precise the correction is bigger. In equation (16) same as the (15) is the state covariance correction step.

The formulations show the recursive nature of the Kalman Filter, which make it more suitable for real time applications such as SLAM. However, in nonlinear cases, linearization or sampling is needed. Sampling is explained in detail in Thrun's work [1], which is called Unscented Kalman Filter. On the other hand, Extended Kalman Filter's solution lies in Taylor Series expansion as explained in detail in the following Section 3.2.2.

3.2.2. Extended Kalman Filter

In nonlinear situations, Kalman Filter is used by extension of linearization process, which is called as Extended Kalman Filter. This filter has two parts as linearization part and filtering part. In the linearization part simple first order Taylor series expansion is used, which needs the first derivatives of the state transition and measurement functions around current mean and covariance value. In the filtering part, the procedure of Kalman Filter is used.

The discrete state transition and measurement functions representation is not applicable in nonlinear situations. Therefore following equations will be used, (17) represents the state transition and (18) represents the measurement functions.

$$x_k = f(x_{k-1}, u_{k-1}, \omega_{k-1}) \tag{17}$$

$$z_k = h(x_k, v_k) \tag{18}$$

Equation (17) is the time update process, which uses nonlinear state transition function f, and equation (18) is the measurement update process, which uses nonlinear measurement function h. These notations of parameters used in function f and h are the same in Kalman Filter. Therefore, the detail explanation is omitted here.

For the given nonlinear model, prediction functions of the next state becomes as (19) and (20).

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1}) \tag{19}$$

$$P_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T \tag{20}$$

Equation (19) is the state prediction, which is derived from the model given in equation (17), and the notations are the same as in equation (17). Equation (20) is the state covariance update, A_k and W_k matrices are the Jacobian matrixes at step k and P_{k-1} and W_k are the noise covariance matrixes at $k-1$ and k respectively. The Jacobian matrixes of state transition and measurement functions, A_k and W_k calculations are given below.

$$A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}}(\hat{x}_{k-1}, u_{k-1}) \tag{21}$$

$$W_{[i,j]} = \frac{\partial f_{[i]}}{\partial w_{[j]}}(\hat{x}_{k-1}, u_{k-1}) \tag{22}$$

Equation (21) is the partial derivative of state transition function with respect to x, and equation (22) is the partial derivative of state transition function with respect to w.

For the given nonlinear model, measurement update functions of state at step k become as in equations (23), (24) and (25).

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + V_k R_k V_k^T)^{-1} \quad (23)$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-)) \quad (24)$$

$$P_k = (I - K_k H_k) P_k^- \quad (25)$$

(23) is the Kalman gain calculation equation, the notation is the same as in (14) except for V_k and H_k. They are the Jacobian matrixes at k, the detail representations are given in equations (26) and (27).

$$H_{[i,j]} = \frac{\partial h_{[i]}}{\partial x_{[j]}}(\tilde{x}_k) \quad (26)$$

$$V_{[i,j]} = \frac{\partial h_{[i]}}{\partial v_{[j]}}(\tilde{x}_k) \quad (27)$$

Equation (26) is the partial derivative of measurement function with respect to x, and equation (27) is the partial derivative of measurement function with respect to w.

3.2.3. Compressed Extended Kalman Filters

Extended Kalman Filter gives a solution for nonlinear cases for with linearization cost. However, its performance is not enough for large-scale environments, which is inevitable in outdoor applications. These types of environments cause a great computational cost for inverse calculations in Kalman gain equation, which is given in equation (23), so N^2 problem occurs. The updates in state transition and measurement equation is usually effects the local area which is independent of the other landmarks. This fact is used in Compressed Extended Kalman Filter to reduce the estimation costs, and gives identical solutions with full EKF state estimation [61].

The idea behind this compressed filter approach is using a local area for standard state and measurement update. When robot quits from this area boundary, it combines this area information with its full map by batch update. This procedure simply reduces the N^2 problem to N_a^2 problem, N_a is the number of landmarks in

active area which is independent of the size of the full map [6]. In Figure 10 the situation is illustrated, the active robot area represented as A, and the stable area represented as B, triangle is robot's current pose and stars are the landmarks.

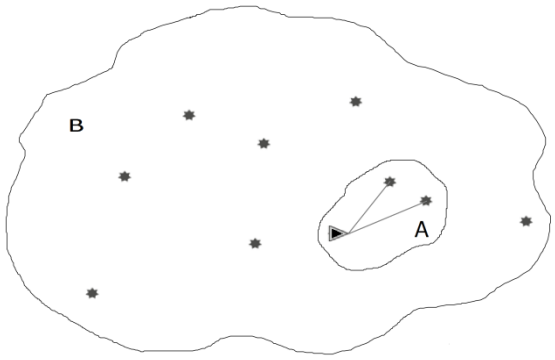


Figure 10 Illustration of Active and Stable Regions of Map

The same nonlinear discrete time dynamic system representation is used for model. The state transition equation and measurement equation are represented in (28) and (29) respectively.

$$x(k+1) = f(x(k), u(k), \omega(k)) \tag{28}$$

$$z(k+1) = h\big(x(k), v(k)\big) \tag{29}$$

These equations have the similar notation as EKF time and measurement update equations represented (17) and (18). However, the different notation is used in this part of the study. Notation (k) is used instead of subscript for iteration number k. The subscript notation is used for sub matrices' notation. $x \in R^n$, $z \in R^m$.

The compressed structure is needed for using this algorithm. Therefore, the state is divided in two parts as in following representation (30).

$$x = \begin{pmatrix} x_a \\ x_b \end{pmatrix} \qquad x_a \in R^{n_a}, \quad x_b \in R^{n_b}, \quad x \in R^n \tag{30}$$

In equation (30) x_a represents the upper part of state matrix, which is the active part, and the x_b represents the lower part, which is kept constant.

The state covariance matrix is divided into two, according to division in equation (30). As a result, following equations are obtained as represented in (31).

$$P = \begin{pmatrix} P_{aa} & P_{ab} \\ P_{ba} & P_{bb} \end{pmatrix} = E\{(\hat{x} - x).(\hat{x} - x)^T\} \in R^{n.n}$$

$$\begin{aligned} P_{aa} &= E\{(\widehat{x_a} - x_a).(\widehat{x_a} - x_a)^T\} \in R^{n_a.n_a} \\ P_{bb} &= E\{(\widehat{x_b} - x_b).(\widehat{x_b} - x_b)^T\} \in R^{n_b.n_b} \\ P_{ab} &= E\{(\widehat{x_a} - x_a).(\widehat{x_b} - x_b)^T\} \in R^{n_a.n_b} \\ P_{ba} &= P_{ab}^T \in R^{n_b.n_a} \end{aligned} \tag{31}$$

It is seen that, the stable part of state x_b has no time or measurement update, when the observations in time period τ between k_1 and k_2 have the following characteristic, as represented in equation (32).

$$\begin{pmatrix} x_a(k+1) \\ x_b(k+1) \end{pmatrix} = \begin{pmatrix} f_a(x_a(k), u(k), \omega_a(k) \\ x_b(k), \omega_b(k) \end{pmatrix}$$

$$z(k+1) = h\big(x(k), v(k)\big) \tag{32}$$

$$\forall\, (x, u, k) / \, k \in \tau$$

Equation (32) shows that, x_a has independent measurement and time update, and w_a, w_b and w_h represent the independent noises. The noise characteristics of the model are represented in the equations (33).

$$E\{\omega_a(k)\} = \bar{0}$$

$$E\{\omega_b(k)\} = \bar{0}$$

$$E\{\omega_h(k)\} = \bar{0}$$

$$E\{\omega_a(k).\omega_a(j)^T\} = \delta_{k,j}.Q_{aa}(k)$$

$$E\{\omega_b(k).\omega_b(j)^T\} = \delta_{k,j}.Q_{bb}(k) \qquad (33)$$

$$E\{\omega_h(k).\omega_h(j)^T\} = \delta_{k,j}.R(k)$$

$$E\{\omega_a(k).\omega_b(j)^T\} = \bar{\bar{0}}$$

$$E\{\omega_a(k).\omega_h(j)^T\} = \bar{\bar{0}}$$

$$E\{\omega_b(k).\omega_h(j)^T\} = \bar{\bar{0}}$$

Equation (33) shows that, the expected values of noises are zero, and their covariance matrices are $Q_{aa,}Q_{bb}, R$ respectively. Moreover, the correlations between them are zero, which means that if the noises are Gaussian these uncorrelated noises are independent.

According to the mentioned model, calculations in the period τ are given in the following part. At the beginning of the period (at k_1) set of auxiliary matrixes $\emptyset, \varphi, \theta, Q_{bb}^*$ are created, which are in the following dimensions as represented in equation (34) and their initial conditions are represented in equation (35).

$$\emptyset, \varphi \in R^{n_a.n_a}, \theta \in R^{n_a}, Q_{bb}^* \in R^{n_b.n_b} \qquad (34)$$

$$\emptyset(k_1) = I, \varphi(k_1) = \bar{\bar{0}}, \theta(k_1) = \bar{0}, Q_{bb}^*(k_1) = \bar{\bar{0}} \qquad (35)$$

In every prediction state calculation in the period τ, standard Extended Kalman Filter calculations of x_a and P_{aa}, and auxiliary matrix calculations are done. These auxiliary matrix calculations are represented in the equations (36).

$$\emptyset(k) = J_{aa} . \emptyset(k-1)$$

$$\varphi(k) = \varphi(k-1)$$

$$\theta(k) = \theta(k-1) \tag{36}$$

$$Q^*_{bb}(k) = Q^*_{bb}(k-1) + Q_{bb}(k)$$

$$J_{aa} = (\partial f_a / \partial x_a)|_{x_a(k)}$$

The following auxiliary matrix calculations are done as represented in equations (37).

$$\emptyset(k) = \big(I - \mu(k)\big) . \emptyset(k-1)$$

$$\varphi(k) = \varphi(k-1) + \emptyset^T(k-1) . \beta(k) . \emptyset(k-1)$$

$$\theta(k) = \theta(k-1) + \emptyset^T(k-2) H_a^T(k-1) S(k-1)^{-1} \tilde{z}(k-1)$$

$$H_a(k) = (\partial h / \partial x_a)|_{x_a(k)}$$

$$\beta(k) = H_a^T(k) . S(k)^{-1} . H_a(k) \tag{37}$$

$$\mu(k) = P_{aa}(k) . \beta(k)$$

$$S(k) = H_a(k) . P_{aa}(k) . H_a^T(k) + R$$

$$\tilde{z}(k) = z(k) - h\big(x_a(k), v(k)\big)$$

Whole state x is updated by using these auxiliary matrixes, when the time period is finished (at k_2), which is called as batch update. The batch update equations are represented in equation (38).

$$P_{ab}(k_2) = \emptyset(k_2) . P_{ab}(k_1)$$

$$P_{bb}(k_2) = P_{bb}(k_1) - P_{ab}(k_1) . \varphi(k_2) . P_{ab}(k_1) + Q^*_{bb}(k_2) \tag{38}$$

$$x_b(k_2) = x_b(k_1) - P_{ab}(k_1) . \theta(k_2)$$

Equation (38) shows that the x_b, P_{bb}, P_{ab} and P_{ba} matrices are only required at the begging and at the end of the time interval τ (at k_1 and k_2). These matrices are kept constant between the interval, and the related information for their updates are calculated and stored by auxiliary matrices $\emptyset, \varphi, \theta$. The auxiliary matrixes have dimensions as $n_a x n_a$, $n_a x n_a$ and n_a respectively. The detailed prove and demonstrations can be found in the given references [61] [6].

This algorithm is very useful in large areas or working with the high frequency rate sensors such as laser sensor. When working with these sensors, the map is updated so frequently, and the computational cost for this batch update will increase a lot. On the other hand, the real time requirement is one of the most important specifications in SLAM applications. Therefore, this algorithm takes attention with its low computational complexity without losing any information. Moreover, batch update of the algorithm can also be done as a background task with a low priority, while updating local map. As a result, these advantages makes Compressed Extended Kalman Filter suitable algorithm for real time outdoor SLAM.

3.3. Global Map Transformation

In this section of study, algorithm details of the techniques mentioned in the Section 2.3.1 are investigated in two parts; global map transformation by using relative measurements and by using maps' overlapping region.

3.3.1. Global Map Transformation by Using Relative Measurements

In this section, global map transformation problem of multi robot SLAM is examined by the use of relative robot measurements. This problem is handled by the mutual relative measurements of robots. The sensor reading, which is composed of bearing and range information, is transferred to other robot and these measurements are processed to calculate the global map transformation matrix between local frames of robots. The following calculations are taken from the Zhou's work [42]. The procedure examined in two parts as relative distance and bearing measurements, transformation between global frames.

3.3.1.1. Relative Distance and Bearing Measurements

Each robot on the map has its own global frame, and they explore and extract the map of the environment on these frames. Robots are represented by R_i and their global frames are represented by $\{G_i\}$ in the following calculations, where $i = 1,2$. The coordinates of R_i, with respect to $\{G_i\}$ is represented as ${}^{G_i}X_{R_i} = [x_i \quad y_i \quad \emptyset_i]^T$. In this representation, x_i is the x-coordinate, y_i is the y-coordinate and $\emptyset_i$ is the direction of the robot pose. The relative position measurement of R_j with respect to R_i is ${}^{i}z_j$, which is given in the equation (39).

$$ {}^{i}z_j = \begin{bmatrix} {}^{i}\rho_m \\ {}^{i}\theta_{jm} \end{bmatrix} = \begin{bmatrix} \rho \\ {}^{i}\theta_j \end{bmatrix} + \begin{bmatrix} \omega_{\,{}^{i}\rho} \\ \omega_{\,{}^{i}\theta_j} \end{bmatrix} \quad i,j = 1,2 \tag{39} $$

In equation (39), ρ is the distance measurement and θ is the bearing measurement of R_i, $\omega_{\,{}^{i}\rho}$ and $\omega_{\,{}^{i}\theta_j}$ are white zero-mean Gaussian noises, whose variance matrices are $\sigma^2_{{}^{i}\rho}$ and $\sigma^2_{{}^{i}\theta_j}$ respectively. Two distance measurements are independent, so precise distance can be calculated with the following equation (40).

$$ \rho_m = \sigma_\rho^2 \left(\frac{{}^{1}\rho_m}{\sigma^2_{{}^{1}\rho}} + \frac{{}^{2}\rho_m}{\sigma^2_{{}^{2}\rho}} \right), \quad \frac{1}{\sigma_\rho^2} = \frac{1}{\sigma^2_{{}^{1}\rho}} + \frac{1}{\sigma^2_{{}^{2}\rho}} \tag{40} $$

These measurements of two robots can be reformed into the following equation (41).

$$ Z = \begin{bmatrix} \rho_m \\ {}^{1}\theta_{2m} \\ {}^{2}\theta_{1m} \end{bmatrix} = \begin{bmatrix} \rho_m \\ {}^{1}\theta_2 \\ {}^{2}\theta_1 \end{bmatrix} + \begin{bmatrix} \omega_\rho \\ \omega_{\,{}^{1}\theta_2} \\ \omega_{\,{}^{2}\theta_1} \end{bmatrix} = Z_t + \omega \tag{41} $$

In equation (41), Z_t is the real value of the measurements and $E[\omega_\rho^2] = \sigma_\rho^2$.

Figure 11 illustrates the geometric relationship between positions and measurements of robots and it is seen that the following equation (42) holds.

$$^{R_1}p_{R_2} = -{}^{R_1}_{R_2}C(\theta)\ {}^{R_2}p_{R_1} \tag{42}$$

$$^{R_j}p_{R_i} = \rho \begin{bmatrix} \cos\ {}^{j}\theta_i \\ \sin\ {}^{j}\theta_i \end{bmatrix} = \rho C(\ {}^{j}\theta_i)e_1, \qquad i = 1,2 \tag{43}$$

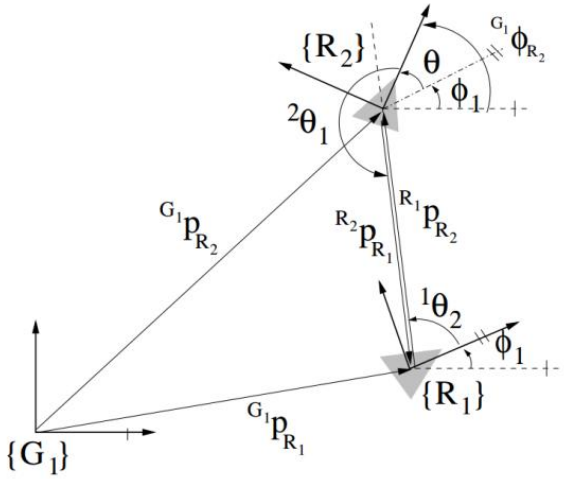


Figure 11 Relationship Between Positions and Measurements of Robots [42]

In equation (42), $^{R_j}p_{R_i}$ is the position of R_i in the local frame of R_j, ${}^{R_1}_{R_2}C(\theta)$ is the angular transformation between $\{R_1\}$ and $\{R_2\}$, and e_1 is the unit vector along the x-axis. By substituting equation (42) into (43) rotational angle between robots' frames can be calculated as mentioned in the following equation (44).

$$\theta = \pi + {}^{1}\theta_2 - {}^{2}\theta_1 \tag{44}$$

The transformation matrix between $\{R_1\}$ and $\{R_2\}$ can be calculated in close form with the angle in equation (44) and the distance in equation (40).

3.3.1.2. Transformation between Global Frames

In this section, by using the bearing and range measurements, transformation between global frames of robots is determined. This transformation enables to represent the state estimate $^{G_1}X_2$ of R_2 with respect to $\{G_1\}$ as in the equation (45).

$$ {}^{G_1}X_2 = h\left({}^{G_1}X_1, {}^{G_2}X_2, Z\right) \tag{45} $$

In equation (45), ${}^{G_1}X_1$ is the state of R_1 with respect to $\{G_1\}$, ${}^{G_2}X_2$ is the state of R_2 with respect to $\{G_2\}$, and Z is the measurement given in the equation (41). The detailed representations of these states are given in the following equation (46).

$$ \begin{gathered} {}^{G_1}X_1 = \begin{bmatrix} {}^{G_1}X_{R_1}^T & {}^{G_1}X_{L_1}^T \dots {}^{G_1}X_{L_i}^T \dots {}^{G_1}X_{L_{n_1}}^T \end{bmatrix} \\ {}^{G_2}X_2 = \begin{bmatrix} {}^{G_2}X_{R_2}^T & {}^{G_2}X_{l_1}^T \dots {}^{G_2}X_{l_j}^T \dots {}^{G_2}X_{l_{n_2}}^T \end{bmatrix} \\ {}^{G_1}X_{L_i} = \begin{bmatrix} X_{L_i} \\ y_{L_i} \end{bmatrix} \in M_1, {}^{G_2}X_{l_i} = \begin{bmatrix} X_{l_i} \\ y_{l_i} \end{bmatrix} \in M_2 \end{gathered} \tag{46} $$

In equation (46), $i = 1 \dots n_1,\ j = 1 \dots n_2$ and $n_1 (n_2)$ is the total landmark count in the map $M_1 (M_2)$ of robot $R_1 (R_2)$.

The rotational transformation of frames from $\{G_2\}$ to $\{G_1\}$ is ${}^{G_1}_{G_2}C$. The rotational transformation of frames from $\{R_2\}$ to $\{G_1\}$ is ${}^{G_1}_{R_2}C$. These transformations can be computed with the following equations (47).

$$ \begin{gathered} {}^{G_1}_{G_2}C(\emptyset) = {}^{G_1}_{R_1}C(\emptyset_1) {}^{R_1}_{R_2}C(\theta) {}^{G_2}_{R_2}C(\emptyset_2)^T \\ => \emptyset = \emptyset_1 + \theta - \emptyset_2 \\ {}^{G_1}_{R_2}C\left({}^{G_1}\emptyset_{R_2}\right) = {}^{G_1}_{R_1}C(\emptyset_1) {}^{R_1}_{R_2}C(\theta) \\ {}^{G_1}\emptyset_{R_2} = \emptyset_1 + \theta \end{gathered} \tag{47} $$

By substituting equation (44) into equation (47) for θ, following equation (48) can be obtained.

$$ {}^{G_1}\emptyset_{R_2} = \emptyset_1 + \pi + {}^1\theta_2 - {}^2\theta_1 \tag{48} $$

In equation (48), $\emptyset_1$ is the orientation of R_1 with respect to $\{G_1\}$.

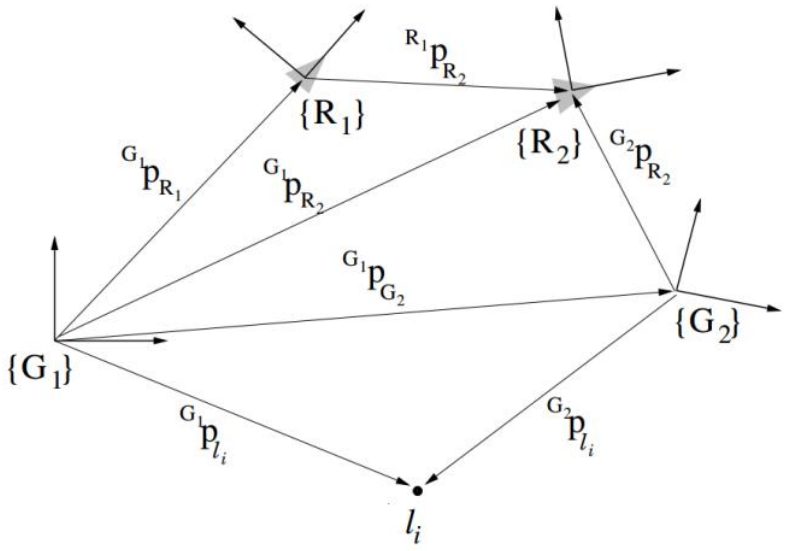


Figure 12 Geometric Relationship Between Global Frames and Landmark Position [42]

In Figure 12, the geometric relationship between global frames and landmark position is illustrated, and it is seen that the following equation (49) holds.

$$\begin{aligned} {}^{G_1}p_{G_2} &= {}^{G_1}p_{R_1} + {}^{G_1}_{R_1}C(\emptyset_1)\,{}^{R_1}p_{R_2} - {}^{G_1}_{G_2}C(\emptyset)\,{}^{G_2}p_{R_2} \\ {}^{G_1}p_{R_2} &= {}^{G_1}p_{R_1} + {}^{G_1}_{R_1}C(\emptyset_1)\,{}^{R_1}p_{R_2} \end{aligned} \tag{49}$$

In equation (49), ${}^{G_i}p_{R_i}$ is the position of R_i with respect to G_i. Similar equation can be written for the position of each landmark $l_i \in M_2$ with respect to $\{G_1\}$ as in equation (50).

$${}^{G_1}p_{l_i} = {}^{G_1}p_{G_2} + {}^{G_1}_{G_2}C(\emptyset)\,{}^{G_2}p_{l_i}, i = 1 \ldots n_2 \tag{50}$$

In equation (50), ${}^{G_1}p_{l_i}$ is the position of landmark l_i in the map M_2 with respect to $\{G_1\}$. By using this transformation landmarks in them map M_2 can be transformed to the global frame of R_1.

3.3.2. Global Map Transformation by Using Map Overlap

In this section, global map transformation problem of multi robot SLAM is examined by the use of overlap between maps. The local map information is transferred to other robots to calculate the global map transformation matrix between local frames of robots. This information is searched with possible transformations, if overlap between maps is found; this transformation can be used as a candidate global map transformation. The best transformation, which gives the perfect overlap between maps, is searched by comparing the overlapping area. This comparison can be done with appropriate heuristic functions such as matched landmark count, similarity metric etc. However, the search space of this transformation is infinite because of the lack of prior knowledge about robot's positions. Therefore, preprocess step can be used for finding similar regions in the map. For this purpose following algorithms; Delaunay Triangulation and similarity metric are examined in the following sections. After this process different search algorithms; Random Sample Consensus, Adaptive Random Walk and Iterative Translation Search are explained in detail. The transformation matrix formulation and derivation is also mentioned in Section 3.3.2.2.1.

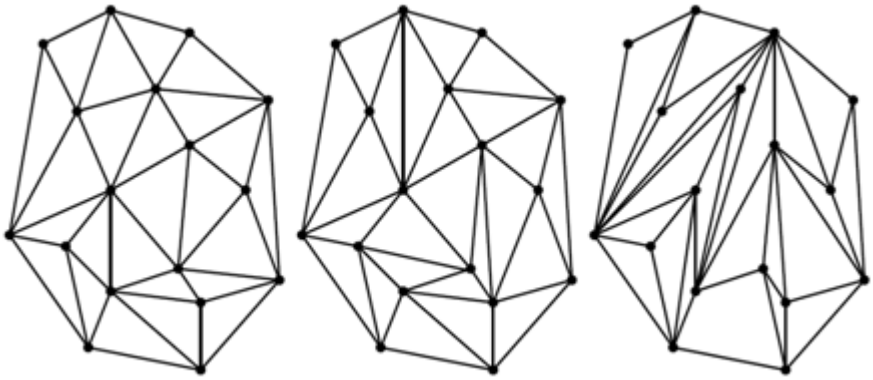
3.3.2.1. Map Structure Similarity

In this section, Delaunay Triangulation and similarity metric calculation techniques, which were used in Cunningham's work [41], are explained in detail.

3.3.2.1.1. Delaunay Triangulation

The complexity of the search space for finding the best transformation matrix between local frames of robots leads to reduction in the input sample set. The geometric feature extraction and their similarities are used for this improvement. In this section, Delaunay Triangulation algorithm [62], which is a well studied geometric feature extraction method is explained in detail. This algorithm takes the coordinates of landmarks as input and calculates the triangles, whose ensure empty circum circle criterion. This criterion means that the circum circle related with the triangle contains no more point than the triangle edges. Moreover, Delaunay Triangle algorithm maximizes the minimum angle of the all triangles,

which avoids the skinny triangles. These properties of algorithm can be seen in Figure 13.



(a) Delaunay Triangles (b) (c) Non Delaunay Triangles

Figure 13 Demonstrations of Possible Triangle Shapes

In Figure 13, (b) and (c) have a more skinny triangles than (a), which shows the possibility of different geometric features.

Delaunay Triangle algorithm gives the same output triangle set for the same input set. However, there exist exceptional cases, in these cases such as rectangular shape, which is illustrated in Figure 14; algorithm is able to produce two different triangles for the same input points.

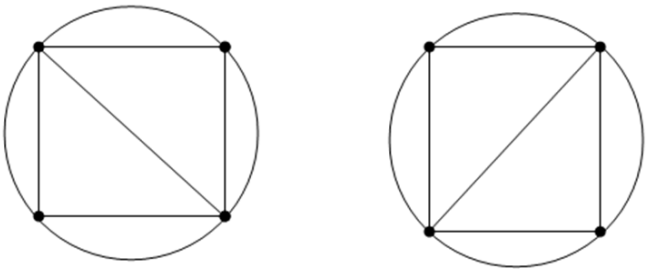
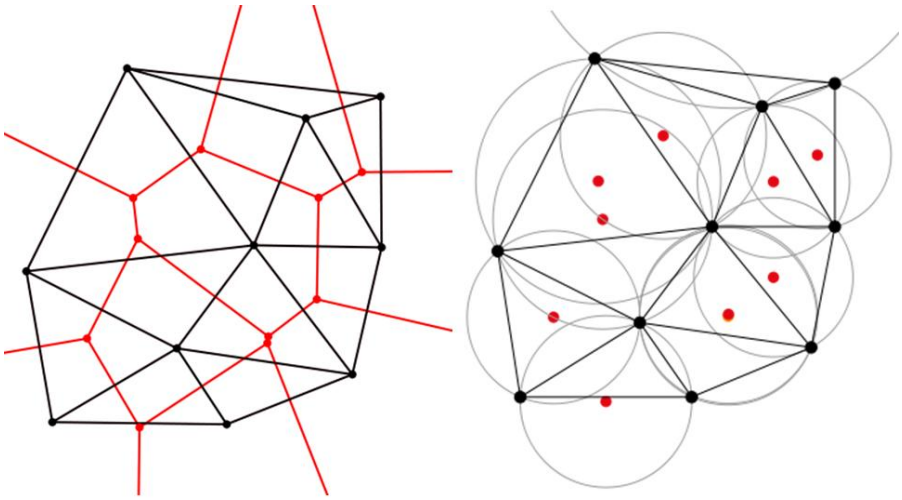


Figure 14 Possible Outputs of DT under Exceptional Case (Rectangular Shape)

The exceptional case presented in Figure 14 has very low possibility in real implementations. The sensor noises and the asymmetric nature of the environment cause the occurrence of the perfect rectangular shape in nature to almost zero.

There exists various types of implementation of the this algorithm in literature such as Guibas and Stolfi's Divide and Conquer [63], Fortune's Sweepline [64], and Bowyer-Watson's Incremental algorithms. The pseudo code of Bowyer-Watson's Incremental Delaunay Triangulation Algorithm is mentioned as an example in the following Table 2.

Delaunay Triangulation is closely related with Nearest Neighbor Graph and Voronoi Diagram [65]. The Nearest Neighbor Graph is a sub graph of the Delaunay Triangulation, because of its empty circum circle property, the closest point to any point lies in the edge of that point in the Delaunay Triangles. Moreover, the center points of the circles of triangles are the joints in the Voronoi diagram. The illustration of this relationship is represented in Figure 5. The dots in (a) show the joints of Voronoi diagram and (b) show the center points of the triangles.



(a) Voronoi Diagram **(b) Delaunay Triangulation**

Figure 15 Illustration of Relationship between Voronoi Diagram and Delaunay Triangulation

Table 2 Pseudo Code of Bowyer-Watson's Incremental Delaunay Triangulation Algorithm [66]

```
function BowyerWatson (pointList)
// pointList is a set of points to be triangulated
1:    triangulation := empty triangle mesh data structure
2:    add super-triangle to triangulation
3:    for each point in pointList do // add all the points one at a time to the
      triangulation
4:       badTriangles := empty set
5:       for each triangle in triangulation do // find the invalid triangle
6:          if point is inside circumcircle of triangle
7:             add triangle to badTriangles
8:       polygon := empty set
9:       for each triangle in badTriangles do // find the boundary of the
      polygonal hole
10:         for each edge in triangle do
11:            if edge is not shared by any other triangles in badTriangles
12:               add edge to polygon
13:      for each triangle in badTriangles do // remove them from the data
      structure
14:         remove triangle from triangulation
15:      for each edge in polygon do // re-triangulate the polygonal hole
16:         newTri := form a triangle from edge to point
17:         add newTri to triangulation
18:   for each triangle in triangulation // done inserting points, now clean up
19:      if triangle contains a vertex from original super-triangle
20:         remove triangle from triangulation
21:   return triangulation
```

The implementation of the Delaunay Triangulation in this thesis, uses Computational Geometry Algorithms Library (CGAL) [67]. The insertion, removal and displacement abilities are available in this library implementation in MATLAB. Triangles can be computed in batch with $O(n * \log(n))$ complexity.

The insertion of a new point is performed by inserting member function of the basic triangulation and performing a sequence of flips for restoring the Delaunay property. The number of flips is $O(d)$ if the new vertex has d degree, for the uniformly distributed points it is $O(1)$ on average. The removal of the point is done by removing and triangulating again, which takes $O(d^2)$ for degree d. The displacement checks whether the triangulation remains planar or not. If triangulation remains, algorithm performs sequence of flips for restoration, which has a complexity of $O(d)$. Otherwise, new point is inserted and the past is removed, which has an $O(n)$ in worst case.

Different implementations are compared with each other in Su's work [68]. His study shows that Dwyer's improvement on the Guibas and Stolfi's Divide and Conquer algorithm is the strongest algorithm in its competitors. However, it is also mentioned that all of these algorithms can be fastest on different dataset.

3.3.2.1.2. Similarity Metric Calculation

In this section, similarity metric used in the Cunningham's work [41] is explained in detail. This similarity metric uses the features of triangles, which are calculated by the use of Delaunay Triangulation algorithm. These features are used for calculating the overlapped regions between two maps and extracted from the Euclidian distances between points. The triangle circumference and area are selected as geometric features for this similarity metric calculation. These feature calculations are represented in the equations (51).

$$f_1^j = a_j + b_j + c_j$$
$$f_2^j = \sqrt{s(s-a_j)(s-b_j)(s-c_j)}, s = \frac{1}{2} f_1^j \quad (51)$$

In equation (51), f_1 represents the circumference of the triangle, and f_2 represents the area of the triangle, and a, b, c are the edges of the triangle.

All features are calculated according to the equation (51) and the correspondence set is calculated by using these features as mentioned in equation (52).

$$C(T_1, T_2) = \{(t_1, t_2) \mid t_1 \in T_1, t_2 \in T_2, S(t_1, t_2) < \tau\} \quad (52)$$

In equation (52), τ represents the threshold value for the similarity match decision. The $C(T_1, T_2)$ represents the correspondence set. The similarity $s(t_1, t_2)$ is calculated as mentioned in equation (53).

$$S(t_1, t_2) = \prod_i e^{\left((f_i^1 - f_i^2)^2\right)} \quad (53)$$

The similarity calculation has errors, because of the geometric similarity of triangles and the selection of the threshold value could be too small or too big. The geometric similarity, which can be similar in different cases, is illustrated in the following Figure 16. The features of two triangles are almost the same but it is seen that they are extracted from different landmarks.

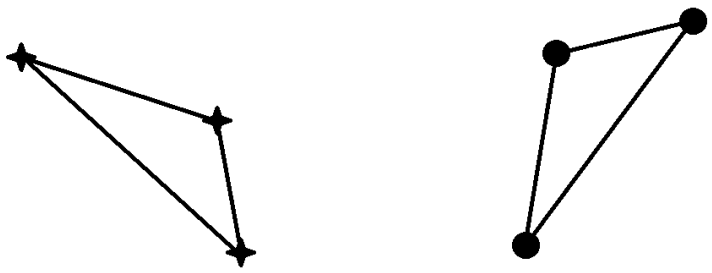


Figure 16 Illustration of Similar Triangles with Different Landmarks

3.3.2.2. Multi Robot Data Association

In this section, transformation matrix formulation with different multi-robot data association algorithms mentioned in Section 2.3.1.2.2 is explained in detail.

3.3.2.2.1. Transformation Matrix Formulation

Transformation matrix, which is searched for map alignment of two local maps, is explained in detail. These maps are built locally and they needs to be transferred to the same reference frame before map merging process. This transfer consists of rotation and translation in two-dimensional space, because of the planar environment. The functional representation of the transformation in Euclidian coordinates is represented in the following equation (54).

$$T_\theta(y) = \begin{bmatrix} \cos\emptyset & -\sin\emptyset \\ \sin\emptyset & \cos\emptyset \end{bmatrix} y + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} \tag{54}$$

In equation (54), φ represents the rotation and t_1, t_2 represent the transformation in x and y coordinates relatively. The related parameter vector can be represented as in the following equation (55).

$$\theta = [\theta_1 \;\; \theta_2 \;\; \theta_3 \;\; \theta_4]^T = [\cos\emptyset \;\; \sin\emptyset \;\; t_1 \;\; t_2]^T \tag{55}$$

By using the equation (55), parameters can be estimated in least square sense, by using the following equation (56).

$$T_\theta(p_1) = [S \quad I_{2x2}]\theta = A\theta, \qquad S = \begin{bmatrix} x_1 & -y_1 \\ y_1 & x_1 \end{bmatrix} \tag{56}$$

In equation (56), S is a skew symmetric matrix and I is the identity matrix. These transformation needs one point (x_1, y_1) to transfer, which means one transfer gives two equations. However, there are four parameters to be calculated, so two independent point transformations are required for close form calculation. Therefore, the cardinality of MSS is $k = 2$. For this calculation, A matrix is augmented as in the following equations (57).

$$\begin{bmatrix} T_\theta(p_1) \\ T_\theta(p_2) \end{bmatrix} = \begin{bmatrix} S^{(1)} & I \\ S^{(2)} & I \end{bmatrix} \theta \tag{57}$$

For performance improvement, M matrix is defined as $M = S^{(1)} - S^{(2)}$, which is Schur complement as represented in equation (58).

$$\begin{bmatrix} T_\theta(p_1) \\ T_\theta(p_2) \end{bmatrix} = \begin{bmatrix} I & I \\ 0 & I \end{bmatrix} \begin{bmatrix} M & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ S^{(2)} & I \end{bmatrix} \theta = \begin{bmatrix} M & I \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ S^{(2)} & I \end{bmatrix} \theta \tag{58}$$

By taking the inverse of the augmented A matrix, the parameters are left on the right hand side of the equation (58), as represented in the equation (59).

$$\begin{bmatrix} I & 0 \\ -S^{(2)} & I \end{bmatrix} \begin{bmatrix} M^{-1} & -M^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} T_\theta(p_1) \\ T_\theta(p_2) \end{bmatrix} = \begin{bmatrix} I & 0 \\ -S^{(2)} & I \end{bmatrix} \begin{bmatrix} M^{-1}\left(T_\theta(p_1) - T_\theta(p_2)\right) \\ T_\theta(p_2) \end{bmatrix} = \begin{bmatrix} \theta_{1:2} \\ \theta_{3:4} \end{bmatrix} \tag{59}$$

This improvement reduces the total algebraic operation of the estimating the model parameter to 12 multiplications and 11 additions. The model estimation's calculation complexity is essential, because it is repeated at every iteration step of search algorithm.

3.3.2.2.2. Random Sample Consensus

The outliers exist after the correspondence set is obtained. Therefore, robust algorithm is needed, while searching for the transformation matrix for global map transformation. Random Sample Consensus is best known for its capability for elimination of outliers and originally used for matching arbitrary point clouds by Fischler and Bolles, in 1981 [69]. It is an iterative algorithm, mostly used for estimating the model of the observed data. It works with a basic assumption, which assumes that, by using a small set of inliers optimal model of the observed data can be found. Algorithm randomly chooses a set of data, and generates a model based on this set. After the candidate model calculation, it is tested on the whole data set and its performance is compared with new generated model.

RANSAC is composed of two essential steps (hypothesize and test), which are repeated iteratively [70].

- In hypothesize step, the Minimal Sample Set (MSS) is selected, which consists of the required number of samples for computing the model parameters. The crucial point here is, the MSS has the smallest number of samples in contrast to other techniques such as least square where the parameter is estimated with all input data.
- In test step, initially computed model is used on the whole data set and the consistent samples are acquired in Consensus Set (CS).

The termination rule of RANSAC is related the probability of finding a better set. If this probability decreases under the threshold or the maximum iteration number is achieved, algorithm terminates with its best model. The detailed representation of these steps is illustrated in the following Figure 17.

The dataset used for model estimation process is represented by $D = \{d_1, \dots, d_N\}$, and the MSS is represented with k. Let the model parameters estimated by RANSAC is $\theta(\{d_1, \dots, d_h\})$, which is estimated by data set $\{d_1, \dots, d_h\}$. The h here is larger than the MSS count (k). The related model is defined as in the equation (60).

$$M(\theta) \stackrel{\text{def}}{=} \left\{ d \in R^d : f_M(d; \theta) = 0 \right\} \tag{60}$$

In equation (60), θ is the parameter vector, and f_M is the smoothing function at zero level it contains the matching parameter set with model M. In real applications, the zero level smoothing is not possible, so the following error calculation in equation (61) and the CS calculation in equation (62) are applied for model extraction.

$$e_M(d; \theta) \stackrel{\text{def}}{=} \min_{d' \in M(\theta)} \text{dist}(d, d') \tag{61}$$

In equation (61), the error e_M is defined as the minimum distance between the data set d and the model space. The distance function selection is done according to the application.

$$S(\theta) \stackrel{\text{def}}{=} \{d \in D: e_M(d;\theta) \leq \delta\} \tag{62}$$

In equation (62), $S(\theta)$ represents the CS with respect given model parameters, and it contains from the samples, which have a smaller error than the given threshold δ.

After the CS is found, error calculation and inliers counting is done and if the inliers ratio is larger than the best model's inliers ratio, current CS's inliers ratio and its related model parameters are stored. The algorithm stop mechanism works in the following manner.

If the probability of choosing the true model from the dataset D is q, then the probability of choosing the set k with at least one outlier becomes $(1-q)$. When the selection of k is iterated over h times, the probability of choosing MSSs with all of them have at least one outlier becomes $(1-q)^h$. The critical point here is this probability goes to zero, if the iteration number h goes to infinity. The real time implementation requirements leads to use acceptable probability threshold ε such that $(1-q)^h \leq \varepsilon$. From this inequality iteration number h can be specify to the following range as in equation (63), in which $\lceil x \rceil$ is used for smallest integer larger than x.

$$h \geq \left\lceil \frac{\log \varepsilon}{\log(1-q)} \right\rceil \tag{63}$$

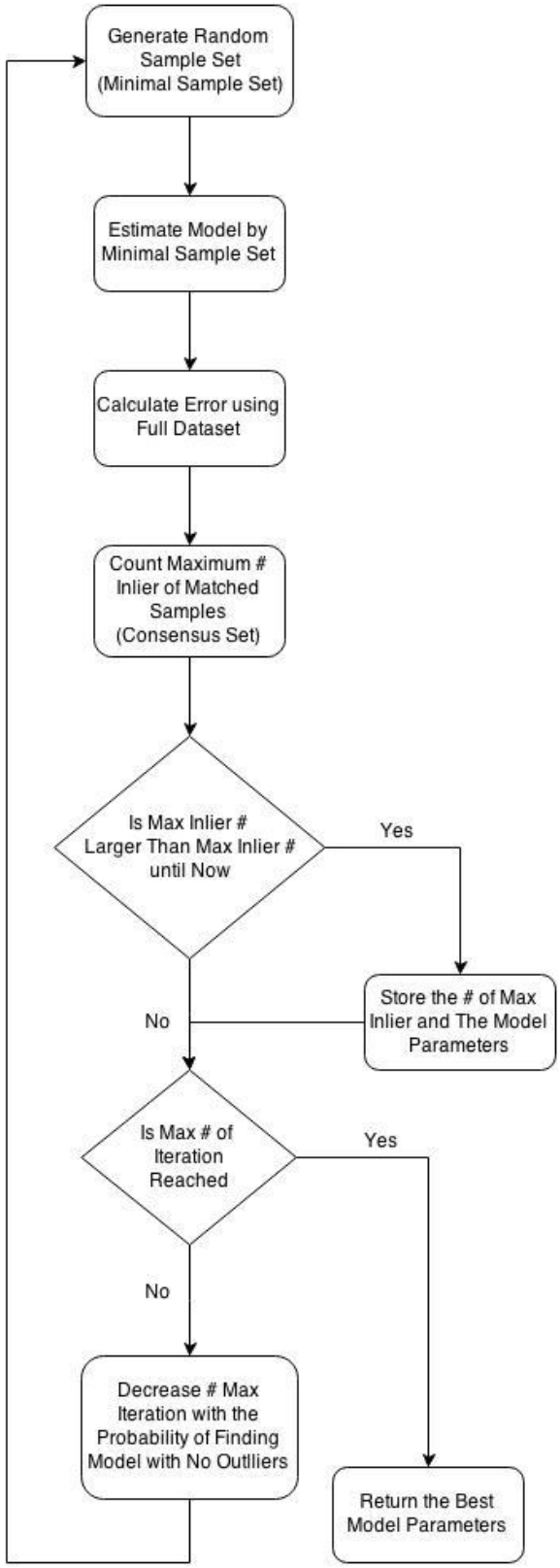


Figure 17 RANSAC Flow Chart

The iteration number h can be set to the minimum integer number by using the using equation (63). The threshold value ε can be determined according to the model, but the probability of choosing the true model q needs to be examined in detail. Under the assumption of equal probability of sample selection and the noise free inliers, which means inliers gives the true model without an error; the following equation (64) can be used for true inliers MSS selection probability calculation.

$$q = \frac{\binom{N_I}{k}}{\binom{N}{k}} = \frac{N_I!\,(N-k)!}{N!\,(N_I-k)!} = \prod_{i=0}^{k-1} \frac{N_I - i}{N - i} \tag{64}$$

In equation (64), N represents the whole dataset number, N_I represents the inliers number and k represents the MSS number. It is seen that if the $N_I >> k$, then the equation (64) is approximated as the following equation (65).

$$q = \prod_{i=0}^{k-1} \frac{N_I - i}{N - i} \approx \left(\frac{N_I}{N}\right)^k \tag{65}$$

In equation (65), there still exists unknown parameter N_I. However, if number of inliers estimated conservatively as $\widehat{N}_I$ which is the largest set of inliers found so far, i.e. $\widehat{N}_I \leq N_I$. Therefore, the following equations hold for this assumption; $q(\widehat{N}_I) \leq q(N_I)$ and $\left(1 - q\left(\widehat{N}_I\right)\right)^h \geq \left(1 - q(N_I)\right)^h$. These results can be used in threshold equation (66).

$$\widehat{T}_{iter} = \left\lceil \frac{\log \varepsilon}{\log\left(1 - q\left(\widehat{N}_I\right)\right)} \right\rceil \tag{66}$$

In equation (66), $\widehat{T}_{iter}$ represents the determined threshold for related $\widehat{N}_I$ estimation.

The overall complexity analysis of RANSAC is done with respect to the hypothesis and test steps. In hypothesis step, the computational complexity of

model estimation is $C_{estimate}\ (k)$, in which from the MSS the parameters are calculated. After the model is instantiated, evaluation of the model over the whole dataset is required. If one sample fitting calculation complexity is $C_{fitting}$, whole step calculation complexity becomes $N * C_{fitting}$. The overall worst-case complexity of RANSAC becomes as the following equation (67), when the all iterations are included.

$$\text{Complexity} = O\left(T_{iter} * \left(C_{estimate}\ (k) + N * C_{fitting}\right)\right) \tag{67}$$

RANSAC is very popular in different areas, such as image processing and SLAM areas. For instance, in image processing area it is used for segmentation of image by feature extraction and parameter fitting [71][72]. On the other hand, in SLAM area, Iser and Wahl used an improved version of RANSAC for loop closing [73] and Cunningham used RANSAC for map merging process with unknown relative pose of robots [41]. In 2006, the 25th anniversary of RANSAC a workshop for examining the variations of RANSAC algorithm is done at International Conference on Computer Vision and Pattern Recognition. The detailed analysis based on different criteria such as speed, accuracy and robustness, is done and the following variations of RANSAC are announced.

The robustness criterion of the RANSAC is examined by Torr [74]. The performance of the RANSAC is directly related with the threshold of the model fitting decision. If the threshold is too small, the inliers can be missed and algorithm time requirement increases, in contrast, if it is too large the outliers can be included in Consensus Set and the model will fluctuate. Therefore, two different versions of RANSAC, M-estimator Sample and Consensus (MSAC) and Maximum Likelihood Estimation Sample and Consensus (MLESAC), which are evaluating the quality of the Consensus Set by using its likelihood, are purposed. Moreover, another improvement is done on the MLESAC by Tordoff, which uses the priori information of samples and called as Guided-MLESAC [75]. Similarly, Chum purposed another improvement by using the same priori information for

deciding the inliers and outliers, which is called as Progressive Sample Consensus (PROSAC) [76].

The speed criterion of RANSAC is improved by Matas and Chum [77], which is called as Randomized RANSAC. Instead of using the entire data set for evaluating the performance of model, randomly selected subsets of the whole data is used. However, this strategy assumes the inliers ratio is large and has less robust performance on data sets with high outlier ratio. Moreover, there exists modifications such as perform on unknown and multiple models. Wang and Suter purposed robust adaptive parameter estimation method for unknown models [78]. For multi model applications, Zulaini's Multi-RANSAC [79] and Toldo's cluster algorithm which, uses J-linkage cluster algorithm [80]. The detailed comparison of these variations of RANSAC can be found in Choi's work [81].

3.3.2.2.3. Adaptive Random Walk

In this section, Carpin's Adaptive Random Walk algorithm is explained in detail. This algorithm can be used for the stochastic search of the transformation matrix between the global frames of robots as an alternative to RANSAC. One map is kept stable while other one is moved by candidate transformations. This movement corresponds to the search of the possible transformations. In this algorithm, randomness is obtained by Gaussian distributed random selector, which is mentioned as Random Selector (RS) in the pseudo code. The adaptive part is obtained by the heuristic function Δ, which accelerates the search in the better and larger overlap in map merging process. Their main motivation of using random walk in the search of the possible transformation space is preventing the algorithm to stick in local maximum with only using adaptive part. The pseudo code of the algorithm is given in the following Table 3.

Adaptive Random Walk algorithm starts the search with the given starting configuration, which is given in the lines 1, 2, and 3. While the iteration number is under the limit value numSteps, new sample is generated and the related heuristic c_s is computed with function Δ, as given in the lines 4, 5, and 6. This generation

step uses a Gaussian distribution with mean μ_k and covariance Σ_k, which are updated with the new accepted sample at step k, as mentioned in the lines 8, 9, and 10. This update is done with the last M values accepted by the algorithm, which is the algorithm parameter. The selection is done with the Random Selector (RS) or with the higher heuristic value c_s, as mentioned in the line 7. If the selection is not done, new sample is discarded and new sample is generated, as mentioned in the line 12 and 4.

Table 3 Pseudo Code of Adaptive Random Walk Algorithm [11]

Adaptive Random Walk	
1:	$k \leftarrow 0, t_k \leftarrow s_{start}$
2:	$\Sigma_0 = \Sigma_{init}, \mu_0 \leftarrow \mu_{init}$
3:	$c_0 \leftarrow \Delta\left(m_1, T_{t_{start}}(m_2)\right)$
4:	while $k <$ numSteps
5:	generate a new sample $s \leftarrow t_k + v_k$
6:	$c_s = \Delta\left(m_1, T_s(m_2)\right)$
7:	if $c_s > c_k$ OR $RS(t_k, s) = s$
8:	$k \leftarrow k + 1, t_k \leftarrow s, c_k = c$
9:	$\Sigma_k \leftarrow \text{Update}(t_k, t_{k-1}, \dots, t_{k-M})$
10	$\mu_k \leftarrow \text{Update}(t_k, t_{k-1}, \dots, t_{k-M})$
11:	else
12:	discard the sample s

The existence of algorithm convergence to best transformation is an important criterion for search algorithms. The following theorem given in the equation (68) ensures this convergence. The proof of the algorithm can be found in Carpin's work [46], which is omitted here.

$$\lim_{k \to +\infty} \Pr\left[\Delta\left(m_1, T_b^k(m_2)\right) \neq \Delta\left(m_1, T_{s^*}(m_2)\right)\right] = 0 \qquad (68)$$

Let $s^* \in S$ an element, which maximizes $\Delta(m_1, T_s(m_2))$, and let $\{T_0, T_1, \dots\}$the sequence of transformations, which is generated by the algorithm in Table 3. T_b^k is the best transformation generated in the first k iterations, which gives the highest value of Δ. This theorem given in the equation (68), only guarantees the convergence, when the iteration goes to infinity.

3.3.2.2.4. Iterative Translation Search

In this section, Saeedi's Iterative Translation Search algorithm [45] is explained in detail. In this algorithm, features are translated with initial translation matrix. Translated features are compared with directions and the features in other map with angular difference smaller than the initial angular threshold are saved. After obtaining possible pairs for all features, matching feature with the smallest Euclidian distance is selected. After this selection, translation matrix is updated with these pairs and the angular threshold is reduced for precise calculation. The algorithm pseudo code is given in the Table 4.

In Table 4, angular threshold ϵ, maximum iteration number $\text{iterations}_{\max}$, translation error threshold J_{thresh}, maps $M_{1,2}$ and norm $n_{1,2}$ of each point in maps $M_{1,2}$ are the parameters of the algorithm. $T_{1,2}[i]$ is used as a container for possible pairs in the maps and T is the output of the algorithm, which gives the best translation at the end of the iteration.

3.4. Feature Based Map Merging

In this section, feature based map-merging algorithms, which are mentioned in Section 2.3.2 are examined in detail. The following methods are examined in the given order; Maximum Likelihood Estimator, Modified M-Estimator, Covariance Intersection Estimator and Orthogonal Gnanadesikan-Kettenring Estimator.

3.4.1. Maximum Likelihood Estimator

Maximum likelihood estimation is a conservative method of combining the observed data, which means that it assumes the input data is good and tries to minimize the error. As mentioned previously, in decentralized architecture

problem becomes the inverse probability $p(\theta|y)$ problem of the given data y. The basic conditional probability equations hold as in given equations (69), which is called as Bayes' theorem.

$$p(\theta, y) = p(\theta)p(y|\theta)$$
$$p(\theta, y) = p(y)p(\theta|y) \tag{69}$$

Table 4 Pseudo Code of Iterative Translation Search [45]

Iterative Translation Search

iteration $\leftarrow 0$
while $J > J_{thresh}$ and iterations $< iterations_{max}$
  $i = 1$
  for $k = 1 \rightarrow$ point number in M_2
    $P_k \leftarrow \{\forall m_1 \in M_1, \text{if } n_1 \text{satisfies } |n_1 - n_2^k| < \epsilon\}$
    if $P_k \neq \emptyset$
      $T_1[i] \leftarrow$ element of P_k that is closest to m_2^k
      $T_2[i] \leftarrow m_2^k$
      $i \leftarrow i + 1$
  $\delta_x \leftarrow \frac{1}{i-1}\left(\sum_{l=1}^{i-1} T_{1x}[l] - \sum_{l=1}^{i-1} T_{2x}[l]\right)$
  $\delta_y \leftarrow \frac{1}{i-1}\left(\sum_{l=1}^{i-1} T_{1y}[l] - \sum_{l=1}^{i-1} T_{2y}[l]\right)$
  $T \leftarrow \begin{bmatrix}\delta_x \\ \delta_y\end{bmatrix}$
  each point in M_2 gets shifted by T
  $J = \sum_{l=1}^{i-1} \left\| T_1^l - T_2^l \right\|$
  reduce ϵ
  iterations $\leftarrow$ iterations $+ 1$

By combining the equations in (69) following equation for conditional density holds as given in equation (70).

$$p(\theta|y) = \frac{p(\theta,y)}{p(y)} = \frac{p(\theta)p(y|\theta)}{p(y)} \tag{70}$$

In equation (70), $p(y)$ is the function of given data, which is called as constant of proportionality, so it can be ignored as in equation (71).

$$p(\theta|y) \propto p(\theta)p(y|\theta) \tag{71}$$

In equation (71), $p(\theta)$ is the prior density of θ, and $p(y|\theta)$ is the likelihood and $p(\theta|y)$ is the posterior of θ. Therefore, it is seen that, the likelihood function converts the prior into the posterior density of θ. The prior density $p(\theta)$ is constant information so it is invariant to the observations, so the following equation (72) holds.

$$\begin{gathered} p(\theta|y) = k(y)p(y|\theta) \\ k(y) = p(\theta)/p(y) \end{gathered} \tag{72}$$

$k(y)$ remains same for given data set of y for all values of θ. It is seen that, without the prior information equation (72) cannot be solved. R.A. Fisher purpose a likelihood notation for solution of this problem. In this theory, parameters of distribution are variables and data is fixed as mentioned in equation (73).

$$L(\theta|y) = k(y)p(y|\theta) \propto p(y|\theta) \tag{73}$$

The likelihood calculation is traditional probability calculation of $p(y|\theta)$, therefore, the observed data is searched for $\widehat{\theta}$, which maximizing the likelihood equation. In multiple scenarios, such as independent observations, the likelihood equation becomes the multiplication of the individual likelihoods as mentioned in equation (74).

$$L = L_1 * L_2 * \ldots * L_N = \prod_{i=1}^{N} L_i \tag{74}$$

In equation (74), $L_i = p(y_i|\hat{\theta})$ instead of minimazing this equation, it is easier to minimize the negative log likelihood as mentioned in equation (75), because of the simplicity of summation over multiplication.

$$\ln L = \sum_{i=1}^{N} \ln p(y_i|\hat{\theta}) \tag{75}$$

Under large sample sets, MLE has the following properties.

- Consistency: If the sample size goes to infinity, MLE converges to the true probability of the estimation.
- Asymptotic Normality: If the sample size goes to infinity, the distribution of MLE tends to Gaussian distribution.
- Efficiency: If the sample size goes to infinity, MLE achieves the lower asymptotical mean squared error, which is called as Cramer-Rao Lower Bound.

The detailed explanations and proofs of MLE properties can be found in the reference [82].

In this thesis, landmark distributions are assumed to be normally distributed. Therefore, the following probability density function (76) is used for parameter estimation process.

$$p(y_1, \dots, y_N|\hat{\theta}) = \frac{1}{2\Pi^{DN/2}|\Sigma|^{N/2}} e^{\left(-\frac{1}{2}\Sigma_1^N (y_n-\mu)^T \Sigma^{-1} (y_n-\mu)\right)} \tag{76}$$

In equation (76), D is the dimension of covariance matrix, Σ is the covariance matrix and μ is the mean value of the distribution. By using sample set and likelihood function, MLE estimates the mean and the covariance parameters of the multivariate normal distribution as follows. Firstly, the logarithm of the distribution of the conditional probability function is taken as in (77).

$$\ln\left(p(y_1,\dots,y_N|\hat{\theta})\right) = \ln\left(\frac{1}{2\Pi^{\frac{DN}{2}}|\Sigma|^{\frac{N}{2}}}\right) + \ln\left(e^{\left(-\frac{1}{2}\Sigma_1^N(y_n-\mu)^T\Sigma^{-1}(y_n-\mu)\right)}\right)$$

$$\ln\left(p(y_1,\dots,y_N|\hat{\theta})\right) = -\frac{DN}{2}\ln(2\Pi) - \frac{N}{2}\ln(|\Sigma|) + -\frac{1}{2}\sum_{n=1}^{N}(y_n-\mu)^T\Sigma^{-1}(y_n-\mu) \quad (77)$$

The equation (77) is differentiated with respect to mean and covariance and equated to zero for the optimal value and the following equations (78) and (79) are derived.

$$\mu_{ML} = \frac{1}{N}\sum_{1}^{N} y_n \quad (78)$$

$$\Sigma_{ML} = \frac{1}{N}\sum_{n=1}^{N}(y_n-\mu_{ML})(y_n-\mu_{ML})^T \quad (79)$$

Detailed proofs of the derivation of the equations (78) and (79), are given in [83].

3.4.2. Modified M-Estimator

Similar to the Maximum Likelihood Estimator calculation steps, modified M-estimator uses the log-likelihood function as mentioned in equation (75). However, the critical difference of this estimator is it uses the covariance matrices of the landmarks, because of the independence between the robots, as mentioned in the equation (80).

$$p(y_1,\dots,y_N|\hat{\theta}) = p(y_1|\hat{\theta})\dots p(y_N|\hat{\theta}) \quad (80)$$

By taking the negative logarithm of the equation (80), similar to the equation (77), the following log-likelihood equation (81) is obtained. In the following equations

matched landmarks estimated mean vectors are represented as y_n and covariance matrices are represented as Σ_n, where N represents the robot number. The true value of mean is represented as μ.

$$
\begin{aligned}
-\ln\big(L(y)\big) &= \frac{1}{2}\sum_{n=1}^{N}\Big((y_n-\mu)^T\Sigma_n^{-1}(y_n-\mu)\Big)+\text{cnst}\\
&= \frac{1}{2}\sum_{n=1}^{N}\left(\Sigma_n^{-1}\left[\mu-\frac{\sum_1^N(\Sigma_n^{-1}y_n)}{\sum_1^N(\Sigma_n^{-1})}\right]^2\right)+\text{const}
\end{aligned}
\tag{81}
$$

By taking the derivative of equation (81) with respect to μ and equate it to zero. The following equation (82) is obtained.

$$
\begin{aligned}
\frac{\partial -\ln\big(L(y)\big)}{\partial\mu} &= \sum_{1}^{N}\Sigma_n^{-1}\left[\mu-\frac{\sum_1^N(\Sigma_n^{-1}y_n)}{\sum_1^N(\Sigma_n^{-1})}\right]^2=0\\
&\left[\mu-\frac{\sum_1^N(\Sigma_n^{-1}y_n)}{\sum_1^N(\Sigma_n^{-1})}\right]=0
\end{aligned}
\tag{82}
$$

The mean, which maximizes the equation (82), is mentioned in the following equation (83) and represented as μ_M.

$$
\mu_M=\frac{\sum_1^N(\Sigma_n^{-1}y_n)}{\sum_1^N(\Sigma_n^{-1})} \tag{83}
$$

The additivity of the statistical information is used for the calculation of the covariance matrix as mentioned in the following equation (84) and combined covariance matrix is represented as Σ_M.

$$
\Sigma_M^{-1}=\sum_{1}^{N}(\Sigma_n^{-1}) \tag{84}
$$

3.4.3. Covariance Intersection Estimator

The matched landmarks estimated mean vectors are represented as y_1, y_2 and covariance matrices are represented as Σ_1, Σ_2. The true values mean vectors are

represented as $\overline{y_1}, \overline{y_2}$ and covariance matrices are represented as $\overline{\Sigma_1}, \overline{\Sigma_2}$ in the following equations.

$$\overline{\Sigma_1} = \mathrm{E}\left[\widetilde{y_1}\widetilde{y_1}^{\mathrm{T}}\right], \quad \overline{\Sigma_2} = \mathrm{E}\left[\widetilde{y_2}\widetilde{y_2}^{\mathrm{T}}\right], \quad \overline{\Sigma_{12}} = \mathrm{E}\left[\widetilde{y_1}\widetilde{y_2}^{\mathrm{T}}\right] \tag{85}$$

In equation (85), $\widetilde{y_1} \stackrel{\text{def}}{=} y_1 - \overline{y_1}$ and $\widetilde{y_2} \stackrel{\text{def}}{=} y_2 - \overline{y_2}$ are the true values of the mean errors and the cross-correlation between random variables is represented as $\overline{\Sigma_{12}}$, which is not known and assumed to be zero.

For ensuring the consistency of the estimation [84], estimator holds the following inequalities (86).

$$\Sigma_1 - \overline{\Sigma_1} \geq 0 \quad \Sigma_2 - \overline{\Sigma_2} \geq 0 \tag{86}$$

By ensuring the inequalities (86), the fused information is guaranteed to be consistent as mentioned in the following inequality (87).

$$\Sigma_{\mathrm{f}} - \overline{\Sigma_{\mathrm{f}}} \geq 0 \tag{87}$$

In inequality (87), $\overline{\Sigma_{\mathrm{f}}} = \mathrm{E}\left[\widetilde{y_{\mathrm{f}}}\widetilde{y_{\mathrm{f}}}^{\mathrm{T}}\right]$ and $\widetilde{y_{\mathrm{f}}} \stackrel{\text{def}}{=} y_{\mathrm{f}} - \overline{y_{\mathrm{f}}}$, $\widetilde{y_{\mathrm{f}}}$ is the true value of the mean error.

This algorithm ensures the given consistency, and uses the convex combination of mean and covariance matrices, which are represented as information (inversed) form. The key point here is using the geometric form of the Kalman Filter equations; mean calculation is represented in following equation (88).

$$\overline{y_{\mathrm{f}}} = \mathrm{W}_1\overline{y_1} - \mathrm{W}_2\overline{y_2} \tag{88}$$

The covariance calculation is represented in the following equation (89).

$$\Sigma_{\mathrm{f}} = \mathrm{W}_1\Sigma_1\mathrm{W}_1^{\mathrm{T}} + \mathrm{W}_1\Sigma_{12}\mathrm{W}_2^{\mathrm{T}} + \mathrm{W}_2\Sigma_{21}\mathrm{W}_1^{\mathrm{T}} + \mathrm{W}_2\Sigma_2\mathrm{W}_2^{\mathrm{T}} \tag{89}$$

In equations (88) and (89), $\mathrm{W}_1, \mathrm{W}_2$ are weight matrices, which can be used for optimizing the estimation with respect to trace or determinant. If the variables are independent, which means that $\Sigma_{12} = 0$, the equations reduces to conventional

Kalman Filter. The geometric interpretation of this algorithm is represented in the following Figure 18.

In Figure 18, the outer ellipses represent the covariance of the random variables Σ_1, Σ_2, and inner ellipses represent the combination of them Σ_f with different values of Σ_{12}. Therefore, if the consistency is satisfied for the intersection area, consistency is also satisfied for every value of cross-correlation even if it is unknown.

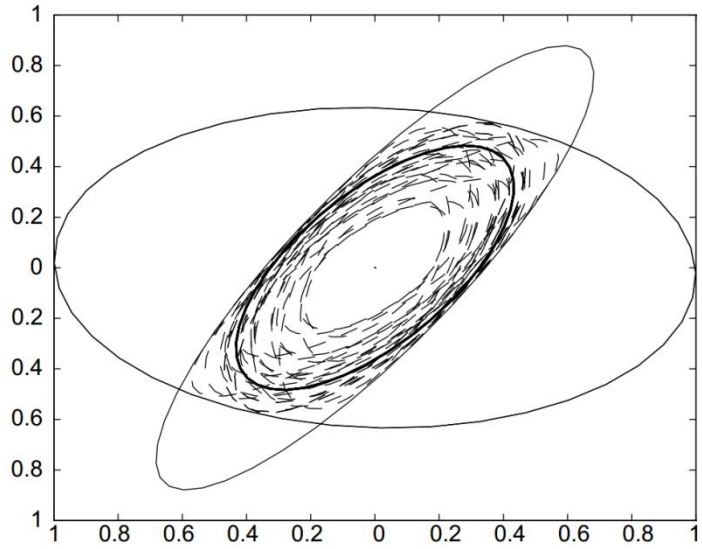


Figure 18 Geometric Illustration of Covariance Matrices [47]

The algorithm satisfies the mentioned consistency by using convex combination of covariance matrices is given in the following equations (90) and (91).

$$\Sigma_f^{-1} = \omega\Sigma_1^{-1} + (1-\omega)\Sigma_2^{-1} \quad (90)$$

$$\Sigma_f^{-1} y_f = \omega\Sigma_1^{-1} y_1 + (1-\omega)\Sigma_2^{-1} y_2 \quad (91)$$

In equation (90) and (91), $\omega \in [0,1]$, and the detailed proof of the equations can be found in the [47]. The choice of ω is depend on the cost function used for optimization, any optimization strategy can be used for search process of ω. The generalized form algorithm for more than two point intersection can be represented as in the following equations (92) and (93).

$$\Sigma_f^{-1} = \sum_1^N \omega_n \Sigma_n^{-1} \tag{92}$$

$$\Sigma_f^{-1} y_f = \sum_1^N \omega_n \Sigma_n^{-1} y_n \tag{93}$$

In equation (92) and (93), ω_n satisfies the $\sum_1^N \omega_n = 1$, which can be used for batch covariance calculations.

3.4.4. Orthogonal Gnanadesikan-Kettenring Estimator

In this section, the class of Orthogonal Gnanadesikan-Kettenring estimator, which is mentioned in the Sequeira's work [53], is explained in detail. This algorithm uses both covariance estimates and actual measurements, which is mentioned in Table 5.

Table 5 Pseudo Code of Orthogonal Gnanadesikan-Kettenring Estimator [53]

OGK Covariance Estimator (Ω is the class parameter)
1: Let $\sigma(.)$ be a standard deviation function applied to its argument
2: Let $X = (X_1, \dots, X_p) \in R^{nxp}$ be the set of n observations, each of dimension p
3: Let $D = \text{diag}\left(\sigma(X_j)\right), j = 1, \dots, p$ and define $Y = XD^{-1}$
4: Compute $U = [u_{jk}] = \begin{cases} \frac{1}{4}\left(\sigma(X_j + X_k)^2 - \sigma(X_j - X_k)^2\right) & j \neq k \\ \sigma(X_j)^2 & j = k \end{cases}$
5: Compute E such that $U = EVE^{-1}$ with V the diagonal matrix with eigenvalues of U
6: Let $Z = YE$ and $A = \Omega E D^{-1}$, with $T = \text{diag}\left(\sigma(Z_j)^2\right), j = 1, \dots, p$ and $\Omega = \text{diag}(\omega_1, \omega_2)$
7: The covariance estimate is $\Sigma = A^T T A$

Orthogonal Gnanadesikan-Kettenring estimator initially scales the input measurements D as mentioned in line 3. Then it computes the initial covariance estimate U, as mentioned in line 4. This initial estimate is used for the new base E calculation as mentioned in the line 5. The scaled data D are projected on this base and new variances are calculated on this frame as mentioned in the line 6. The Ω matrix in line 6 is the input parameter of the estimator and affects the scale of the resultant covariance estimate. In the final step of estimator data are back projected onto its original frame, as mentioned in the line 7.

This algorithm estimates the covariance by the use of the distance between measurements. For instance, the distant data on the smaller variance valued axis, are considered as outliers and eliminated by the algorithm.

3.5. Map Performance Evaluation

In this section, map performance evaluation techniques, mentioned in Section 2.4 are examined in detail.

3.5.1. Normalized Estimation Error Squared Consistency Test

In this section, Normalized Estimation Error Squared (NEES) consistency test is examined in detail. This test simply estimates the error between estimated feature position and ground truth position of feature as mentioned in equation (94).

$$\epsilon_{l_i} = \left(x_{l_i} - \bar{x}_{l_i}\right)^T {\Sigma_{l_i}}^{-1} \left(x_{l_i} - \bar{x}_{l_i}\right) \tag{94}$$

In equation (94), ϵ_{l_i} is the error of estimated feature position x_{l_i} and Σ_{l_i} is the covariance of i^{th} landmark l_i. $\bar{x}_{l_i}$ is the true values of feature position.

In χ^2 distribution degrees of freedom is taken as the dimension of feature positions, which is two. This calculation is gated for acceptance test as given in the following equation (95).

$$\frac{1}{K}\sum_{k=1}^{K} {\epsilon_{l_i}}^k < \gamma_2 \tag{95}$$

In equation (95), K is the Monte Carlo run count of the error ϵ_{l_i}, and γ_2 is the acceptance gate, which accepts the 99% correct association for value 9.21.

3.5.2. Area of Covariance Ellipse Metric

In this technique, area of ellipse used as an evaluation metric is explained in detail. These ellipses are obtained from the covariance matrices of the features in the map. This method is applied to maps, which are the resultant maps of different map merging algorithms.

The area of ellipses A, calculation is given in the following equation (96).

$$A(\Sigma_{l_i}) = \pi r_1 r_2 \tag{96}$$

In equation (96), r_1 and r_2 represents the biggest and smallest radius of the ellipse, and Σ_{l_i} represents the covariance matrix of i^{th} landmark l_i. In the covariance ellipse these values are the square roots of eigenvalues λ_1, λ_2, and the area simply becomes the determinant of the matrix as given in the following equation (97).

$$A(\Sigma_{l_i}) = \pi\sqrt{\lambda_1\lambda_2} = \pi\sqrt{\det(\Sigma_{l_i})} \tag{97}$$

CHAPTER 4

SIMULATOR AND MAPS

4.1. Simulation Environment

In this section, the simulator used in the experiments is explained in detail. The main purposes of implementing this simulator are

- defining different maps,
- planning different trajectories for different robots,
- representing the local maps of robots,
- represents the merged global map,
- defining different ranges for robot sensors,
- evaluating of different combining methods.

The simulation code for one robot is taken from the Open-SLAM community. This one robot simulator program uses the Compressed Extended Kalman Filter based SLAM implementation and it is implemented by Z. Haiqiang [85]. This simulator code is used for one robot SLAM purpose. This simulator inputs are the landmark set and the waypoints of the robot. The SLAM simulator uses the motion model for trajectory and a sensor model for observations. For easy understanding, snapshot of the simulator is given in the following Figure 19.

In Figure 19, the true robot's position is represented by blue colored triangle; the estimated value of the robot's position is represented by red colored triangle with red colored ellipse. The landmarks' true positions are represented by blue colored stars and estimated positions are represented by red colored pluses with red

covariance ellipses. The covariance ellipses are drawn by 2.448 sigma values for cumulative distribution function, which correspondence in 0.95 confidence region. The true trajectory of the robot is represented by the blue line, and the estimated trajectory of the robot is represented by the red line. The trajectory planning of the robot is given in detail in the following section.

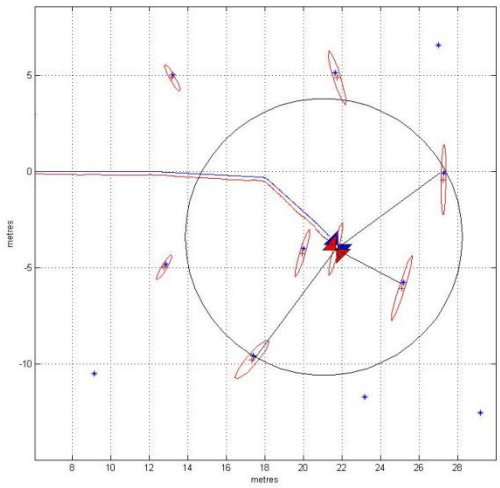


Figure 19 Simulator Environment Screenshot

Defining different maps, planning different trajectories, and representing the merged map are the requirements of the implemented simulator. The implemented simulator supplies a graphical user interface and uses the Haiqiang's simulator code. The snapshot of the implemented simulator is represented in Figure 20.

In Figure 20, the simulator user interface is represented. In the plots, named as Robot 1 and Robot 2, different trajectory SLAM results, resultant triangles and merged map results are plotted.

The simulator searches in the "DataSets" folder and lists the subfolders in the dataset popup menu. These folders categorize the different datasets by name. After the dataset selection, simulator searches for the "landmarks.mat" file, which is used as reserved name. This file contains the x and y coordinates of the landmarks on the map. Simulator contains 3 different datasets, which are "Central Park", "Victoria Park" and "Simulated Park". Users are able to use these

predefined datasets, and they are able to define new datasets. “Manual Entry” option in the dataset menu activates the new popup dialog box named as “Map Dimensions”, users are responsible for entering the related x and y dimensions of examined map. The box is illustrated in Figure 21.

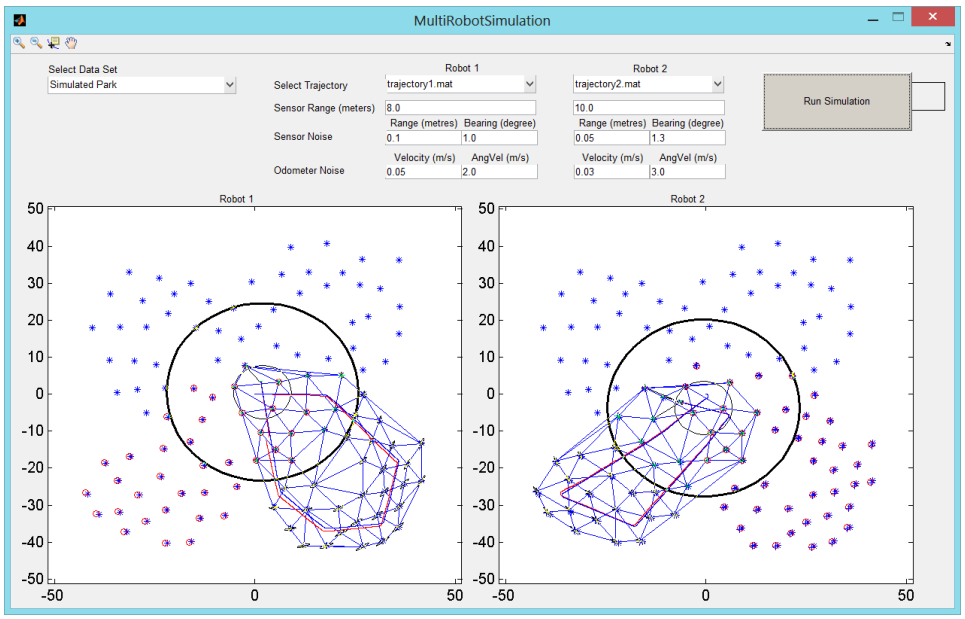


Figure 20 Multi Robot Simulator Environment Screenshot

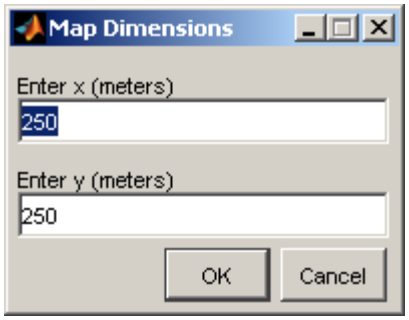


Figure 21 Map Dimensions Popup Dialog Box

After this step, a new popup window is opened for manual landmark entry. By left mouse clicks on the map, simulator stores the landmarks. This window is represented in Figure 22 for easy understanding.

After the landmark entry, by right click on the mouse button saving step is started. In this step, saving the dataset under the “DataSets” folder is needed and the

“landmarks.mat” is the reserved name for saving a new dataset correctly. The saving window is illustrated in Figure 23.

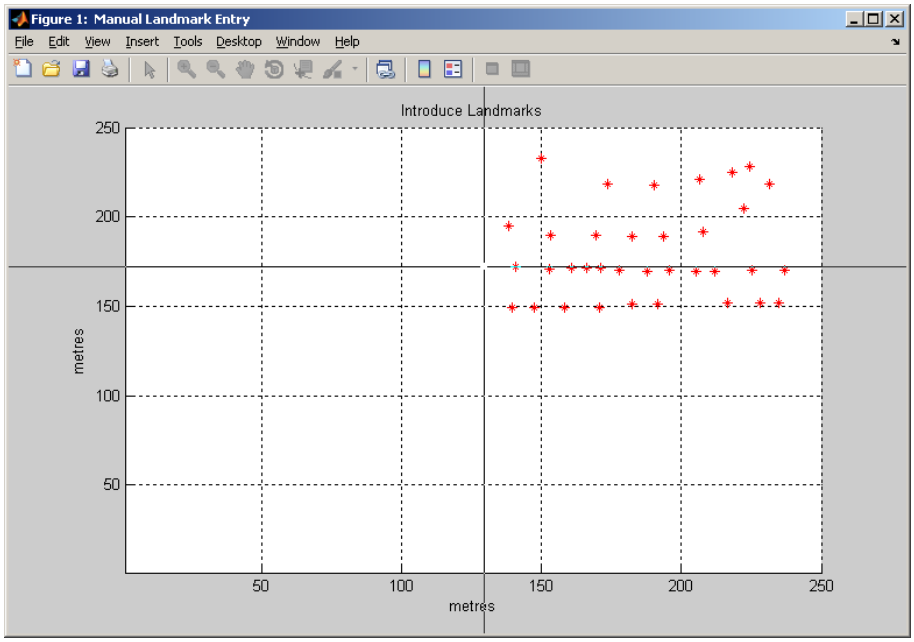


Figure 22 Manual Landmark Entry View

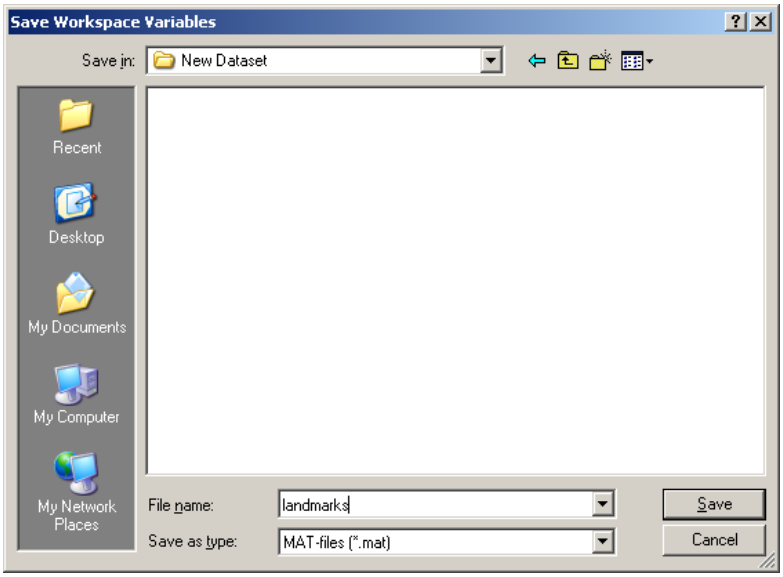


Figure 23 Saving Window View

After the selection of the dataset, the loaded maps are plotted on the Robot1 and Robot2 axes. The trajectory popup menu lists the ".mat" files in the selected

dataset folder for listing predefined trajectories. Users are able to select these trajectories, and they are able to define new trajectories for robots. "Manual Entry" option in the trajectory menu activates the trajectory defining ability. By left mouse clicks on the map, simulator saves the trajectory points sequentially. This scenario is represented in Figure 24.

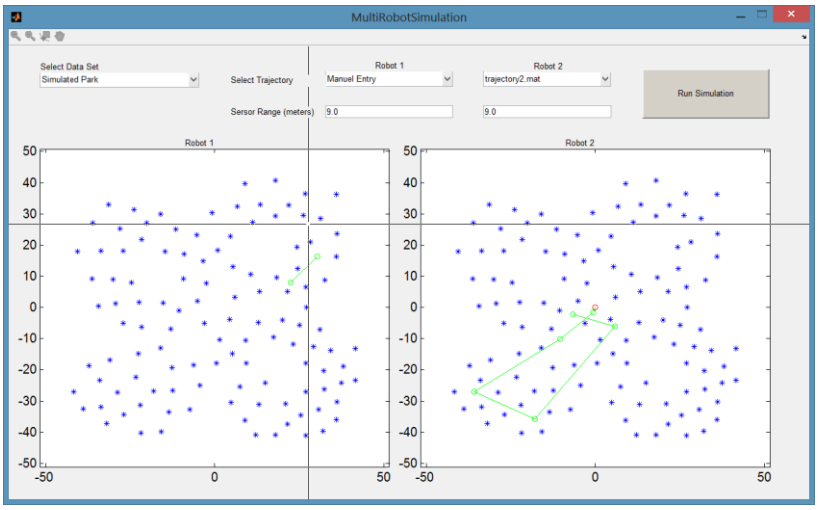


Figure 24 Manuel Trajectory Planning Ability

In Figure 24, crossed line intersection shows the next point of the trajectory. The circles represent the already placed trajectory points.

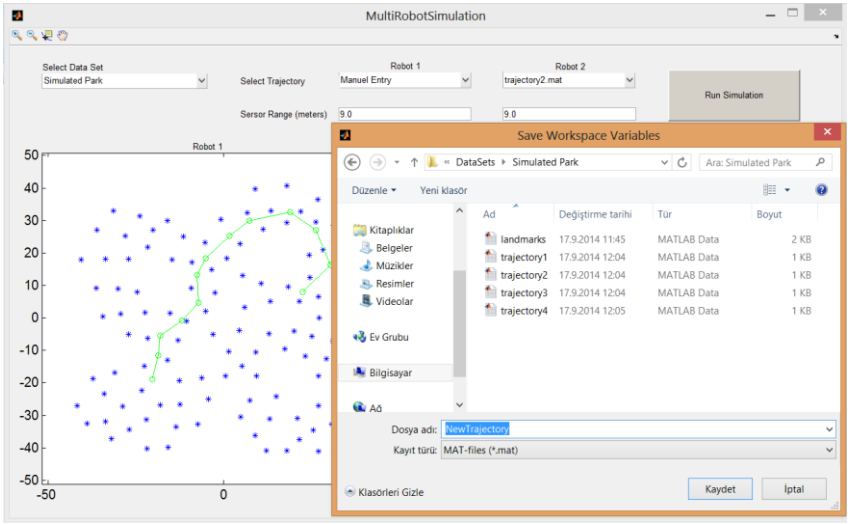


Figure 25 Trajectory Saving Ability

In Figure 25, planned trajectory is named as "NewTrajectoy" and it will be saved under the "DataSets\Simulated Park" directory.

The trajectory planning process can be finished by right mouse click. In this step, simulator gives opportunity to the user for saving this trajectory. In the following Figure 25, the finishing procedure is represented.

This saving ability gives opportunity to use this trajectory for other robot, also stores this information permanently for other experiments. Changing sensor range of the robots is another ability of the simulator. Different sensor ranges can be used in the simulation.

After setting map, trajectory and sensor ranges for all robots, by clicking on "Run Simulation" button simulation starts. Robots follow the given trajectory and sense the environment by the given sensor range limitation. When the individual SLAM process finishes, robots local maps are shared for map merging purpose. These shared map information is used for global map calculation and the resultant map of the robots are plotted with their local maps.

4.1.1. Random Map Generator

The landmark set generation is done by map generator algorithm, which is implemented in MATLAB environment. The generator takes the point number, range and distance limitation. Generator tries to generate number of points in the given range; if the distance limitation is also required, it also put minimum distance between points. The following represents the different generated maps by the random map generator.

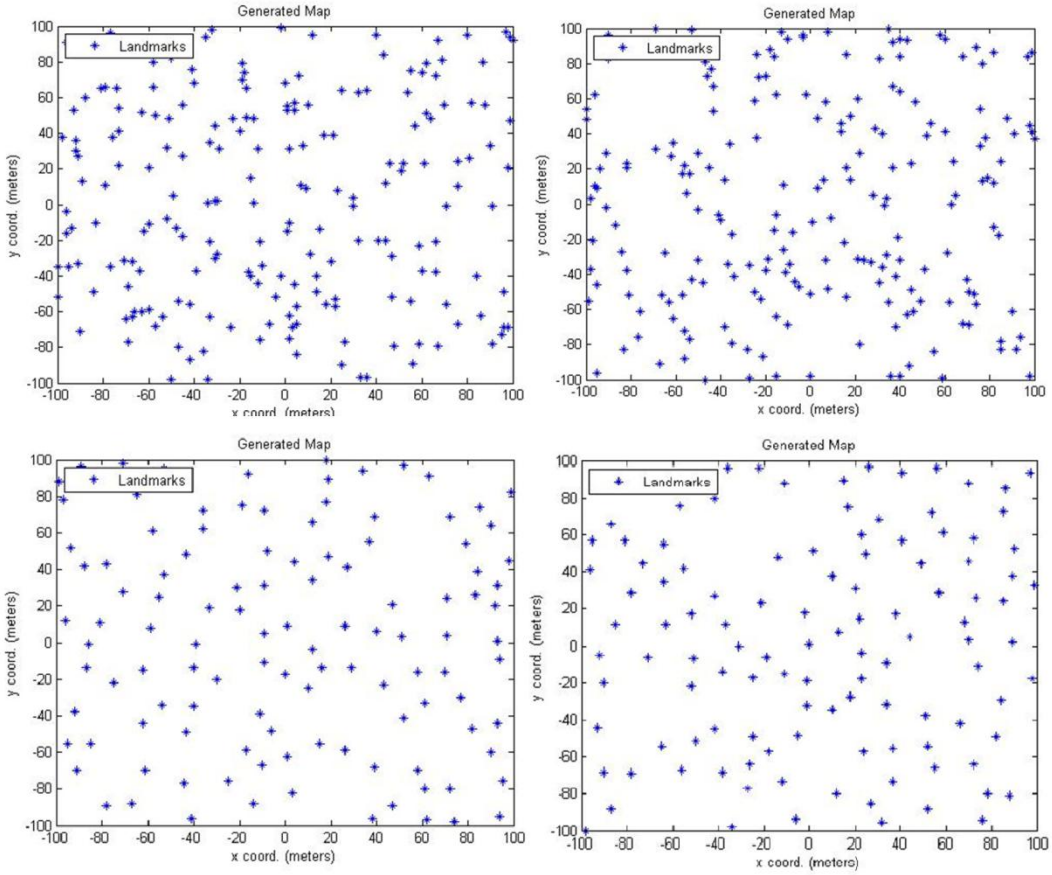


Figure 26 Illustration of Generated Maps with and without Distance Limitation (without Distance Limitation and with 10 meters Distance Limitation)

In Figure 26, (a) and (b) represents the random nature of the generated maps, (c) and (d) represents the distance limitation effect on the generated maps.

4.1.2. Trajectory Planning

The simulator uses the landmarks and waypoints for simulation, which is given in the begging of the SLAM process. The waypoints are in two dimensional array structures, which consist of the ordered points. These points are visited by robot in the given order. The important part of the path planning is that, is the given trajectory possible for the robot specifications. The robot motion model is setup to be like a Pioneer3-AT robot, which is skid steering. Therefore, the rotational speed of the robot is limited by a configurable constant, which is used as 30

degree per second (0.5236 meters per second). The transition speed of the robot is used as configurable constant, which is set to 0.3 meters per second. The flow chart of the algorithm is given in the following Figure 27.

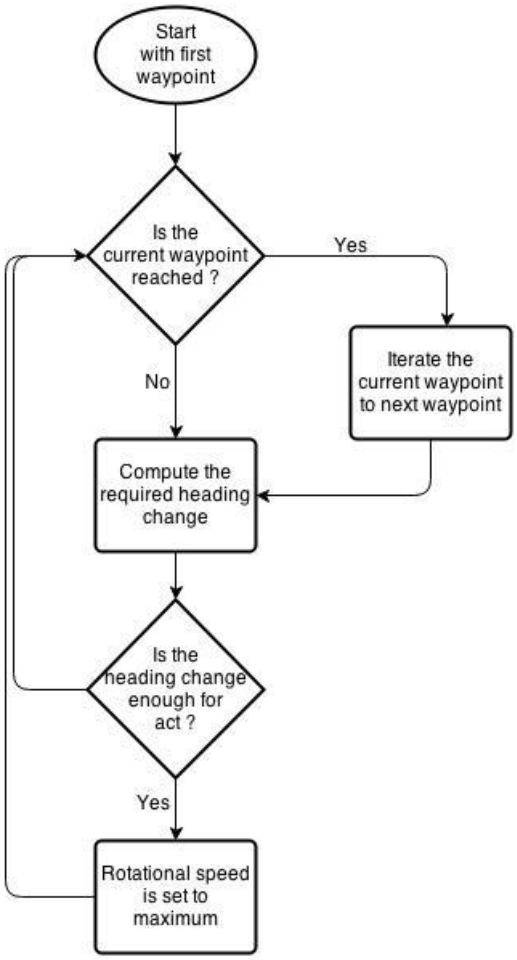
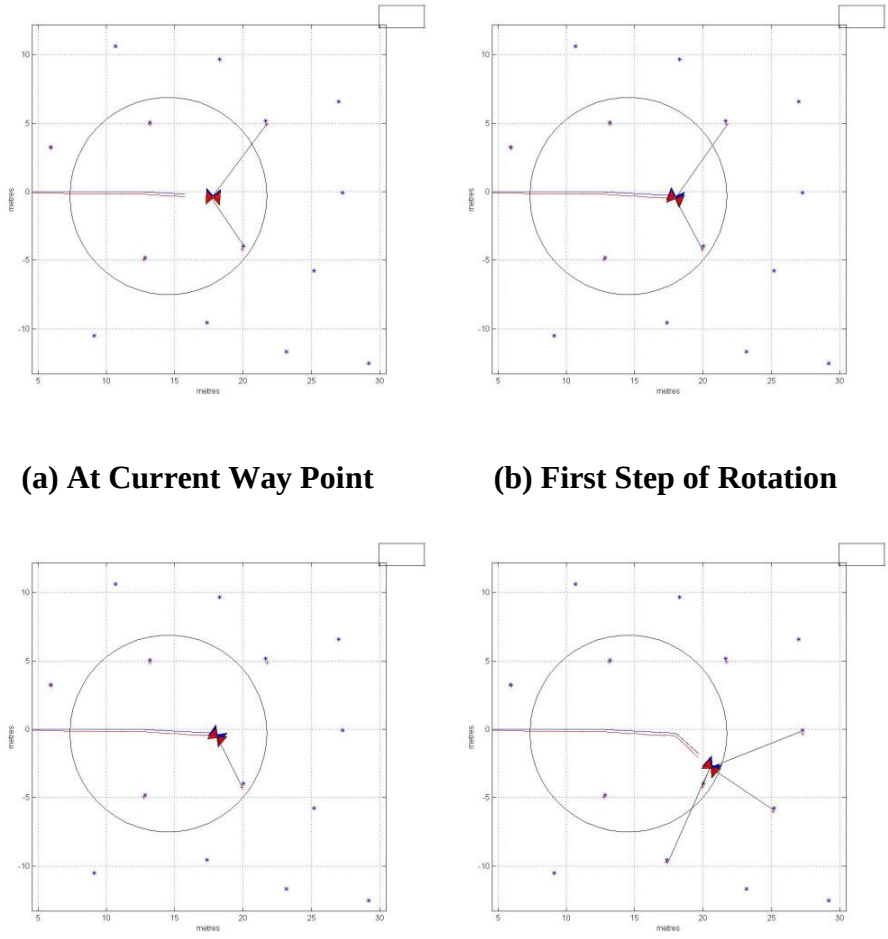


Figure 27 Flow Chart of the Trajectory Planning Algorithm

In trajectory planning part of the simulation, true value of the robot position is used for understanding the robot's current location relative to the next waypoint. If robot is close enough to next waypoint, the next waypoint is updated and the required rotational speed calculation starts. The current location and the next waypoint are used for calculation of the required heading change of the robot. If the required heading change is larger than the maximum rotational speed, the current rotational speed is set to maximum, otherwise the requirement is considered as zero and nothing is done.

The rotational motion of the robot is illustrated in the following Figure 28, for easy understanding of the systematic rotational motion.



(a) At Current Way Point **(b) First Step of Rotation**

(c) Second Step of Rotation **(b) Heading to Next Way Point**

Figure 28 Rotational Motion of Robot

4.1.3. Motion Model

The robot motion model is setup to be like a Pioneer3-AT robot, which has four wheels and four motors, skid steer robot. The picture of the original robot is shown in the following Figure 29.

Figure 29 Picture of Pioneer3-AT Robot

Pioneer3-AT has 0.7 meters per second maximum forward and backward speed and 140 degree per second rotational speed. In our simulator, about half and quarter of the maximum specifications are used for maximum values. The motion model of the given robot without rotational velocity is given in the following equations (98).

$$x = x + \begin{bmatrix} V * dt * \cos(x(3)) \\ V * dt * \sin(x(3)) \\ 0 \end{bmatrix} \tag{98}$$

In equation (98), V represents the transition speed and dt represents the time difference for every step. If the rotational is calculated by the algorithm explained in Figure 27, the following equation (99) is used for the next true value of the robot's position.

$$x = \begin{bmatrix} x(1) + \frac{V}{W} * \left(\sin(x(3) + W * dt) - \sin(x(3))\right) \\ x(2) + \frac{V}{W} * (\cos x(3) - \cos(x(3) + W * dt)) \\ \text{pi\_to\_pi}(x(3) + W * dt) \end{bmatrix} \tag{99}$$

In equation (99), W represents the rotational speed and pi_to_pi() represents the MATLAB function, which is used for taking mod of the input angle with respect to pi.

The iterated robot position by using the true position of the robot is noised with the configurable zero mean normally distributed noise, which is called as control noise. The control noise consists of transitional velocity noise and rotational velocity noise. Transitional velocity noise is generated with 0.03 sigma value, and rotational velocity noise is generated with 0.0524 sigma value. The generated noises are used to disturbed the true values and feed to the CEKF-SLAM algorithm prediction step.

4.1.4. Observation Model

Observation model is setup to be like LMS 200, which is a bearing and range sensor. LMS 200 operates with principle of laser light velocity and time difference between the reflection and the transmission. The picture of LMS 200 is represented in the following Figure 30, for easy understanding.

Figure 30 Picture of LMS 200

While the robot moves simulator controls the landmarks can be observed according to the sensor specifications. The sensor observation frequency is set to five times to the control input iteration time, which is used as 0.5 seconds. Whenever the given landmark set is in the range of sensor, the observation model is used for calculating the bearing and range of landmark. The range of sensor is configurable value, which is set to 8 meters as default value. The scanning range of the sensor is 180 degree, and the scanning illustration is represented in the following Figure 31.

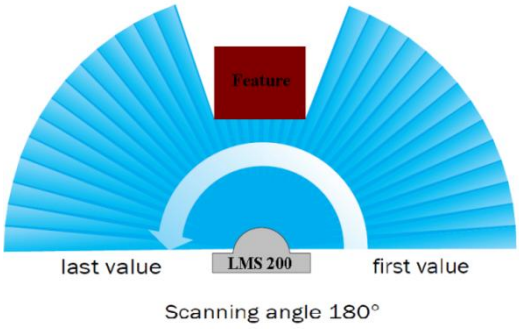


Figure 31 Illustration of the Scan Model of LMS 200 on Top View

Simulator controls the landmark set according to the given specifications below, and extract the visible landmarks within the semi-circular field of view of the sensor. In control procedure, bounding box test, bounding line test and bounding circle test are applied in the given order for efficiently searching for the visible landmarks. The simple explanations are given for the applied test procedures.

4.1.4.1. Bounding Box Test

In bounding box test step, the maximum observation range is compared with the x and y coordinates distances between robot and tested landmark. If one the distance is larger than the maximum observation range, testing procedure is failed and the landmark is considered as invisible, otherwise the following test procedures are applied. The following Figure 32, illustrates the bounding box test procedure.

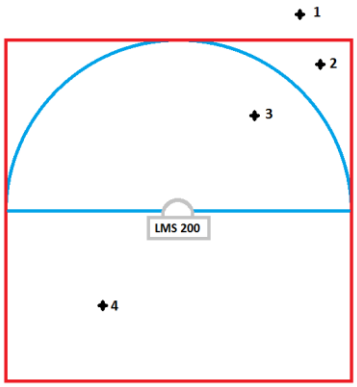


Figure 32 Bounding Box Test Illustration

In Figure 32, rectangle represents the bounding box acceptance borders, semi circle represents true scanning range and the pluses show the landmarks. According to the bounding box test, first landmark is eliminated and second, third and fourth landmarks are passed to the next test procedure.

4.1.4.2. Bounding Line Test

In bounding line test step, the eliminated landmarks are tested whether they are in front of the robot of not. Bounding line test is done by using the heading angle of robot and the x and y coordinate distances between robot and tested landmark. X and y moments of the landmark is calculated by multiplying the x coordinate distance with cosine of heading angle, and y coordinate distance with sine of the heading angle. The moments are summed and checked if it is larger than zero. If the summation is large than zero, it means that landmark is in front of the robot. The following Figure 33 illustrates the bounding line test procedure.

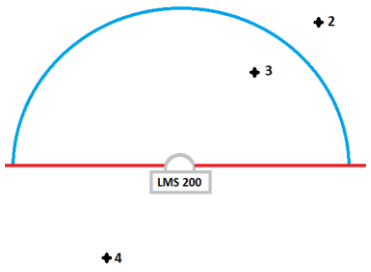


Figure 33 Bounding Line Test Illustration

In Figure 33, line represents the bounding line acceptance borders, semi circle represents true scanning range and the pluses show the landmarks. According to the bounding line test, fourth landmark is eliminated and second and third landmarks are passed to the next test procedure.

4.1.4.3. Bounding Circle Test

In bounding circle test step, the eliminated landmarks are tested whether they are in circular range sensor of not. Bounding circle test is done by using the x and y coordinates distances between robot and tested landmark. The square root of sum of the square of distances is checked if it is larger than maximum observation

range. If this final test is passed, it means that landmark is in the range of the true sensing region and it is visible to sensor. The following Figure 34 illustrates the bounding circle test procedure.

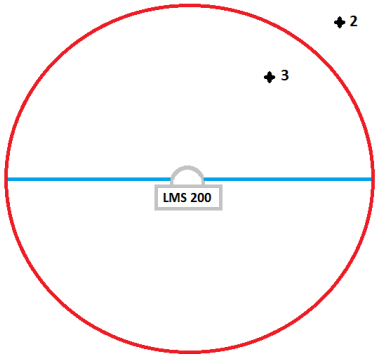


Figure 34 Bounding Circle Test Illustration

In Figure 34, line represents the bounding circle acceptance borders, upper half of the semi circle represents the true scanning range and the pluses show the landmarks. According to the bounding circle test, second landmark is eliminated and third landmark is accepted as visible landmark.

After the visible landmarks are found, they are ready for bearing and range calculation. That calculation is done simulator for every visible landmark, before adding the observation noise. The calculation of bearing and range is mentioned in the following equation (100).

$$z = \begin{bmatrix} \sqrt[2]{dx^2 + dy^2} \\ \text{pi\_to\_pi}(\text{atan2}(dy, dx) - \emptyset) \end{bmatrix} \quad (100)$$

In equation (100), dx and dy represents the x and y coordinate distances, $\text{pi\_to\_pi}(), \text{atan2}()$ are MATLAB functions and $\emptyset$ is the heading angle of the robot. $\text{pi\_to\_pi}()$ takes the mod of the given angle with respect to pi, and $\text{atan2}()$ gives the inverse tangent value of the given distances.

The sensor measurement is disturbed with configurable zero mean normally distributed noise, which is called as observation noise. Observation noise consists

of bearing and range noise. Range noise is generated with 0.05 meter sigma value, and bearing noise is generated with 0.0175 radian sigma value. The generated noises are used to disturbed the true values and feed to the CEKF-SLAM algorithm update step.

4.2. Maps and Trajectories

The comparison of the performances of the map merging algorithms requires different test cases. These cases can be obtained by changing the algorithm parameters or running algorithms with different inputs. Changing algorithm parameters can be obtained by taking and setting different values by the user interface of the simulator. On the other hand, running different algorithms with different inputs, needs more detailed procedure. These inputs of the algorithm are maps and trajectories. The implemented simulator provides an easy graphical interface for creating different maps and trajectories. The detailed explanation of this procedure can be found in the section 4.1. In this section, the map and trajectory selection criteria are mentioned.

The different algorithm parameters and different inputs enable the algorithms performances can be examined in detailed. However, using large test spaces such as too many different maps and trajectories can also mislead to wrong evaluations and waste of time. Therefore, real world datasets are very useful for obtaining more realistic evaluation conditions. In this section, the extraction of different landmark data sets and the related trajectory planning methods used in this study and the reason for choosing these maps and trajectories are explained in detailed.

4.2.1. Extraction of Different Landmark Datasets

The landmark data sets consist of the extracted landmarks' x and y coordinates. These coordinates represent the positions of the features on the map. These positions can be extracted from the sensor readings in real world test scenarios. On the other hand, these positions can also be obtained by using random map generators or by entering manually. The extraction of positions from sensor reading method can be obtained from the real world experiments. These

experiments need real robots, sensors and stable environment for consistent data set. This scenario is costly and time consuming, so it is out of the scope of this study.

In the experimental part of the thesis, randomly generated maps and manually entered maps are used. However, generated maps are randomly distributed maps and they enforce the algorithms computational costs increase. These maps are used for extreme test scenarios, but still more maps that are realistic are required for more meaningful comparison criteria. Therefore, manual entry ability of simulator is used for dataset generation. By using this ability, more structured maps are obtained and the real world maps are used as reference maps for real world data set. The real world data set extraction has two main procedures, obtaining maps of meaningful area and manual landmark labeling.

4.2.1.1. Obtaining Meaningful Real World Map

The main reason for using real world data is obtaining more realistic data set than the randomly distributed data sets. Therefore, more structured areas than distributed areas are examined. For instance, parks are the popular places for outdoor SLAM, because of similar structure of trees. This similarity is used for landmark extraction and data association algorithms in SLAM. Moreover, the tree distribution of the parks generally has a structure. For instance, trees occur on the sides of the paths in the park. Therefore, the algorithms used for understanding the similarity between local maps of robots, perform better by using this structured distribution. Without this distribution, triangles have similar properties, and this similarity leads algorithm to match false pairs of landmarks. Therefore, using the structured area is one the reason for selection of parks. The other criterion is the popularity of the parks in SLAM literature and dataset availability.

Victoria Park and Central Park are selected as reference real world maps in this study. Victoria Park is one the most popular map for SLAM community and its dataset is available. Thrun's study on Victoria Park [1] is used for extracting

Victoria Park dataset. In his study, the extracted landmarks are already labeled by SLAM algorithm. This labeled map is represented in Figure 35.

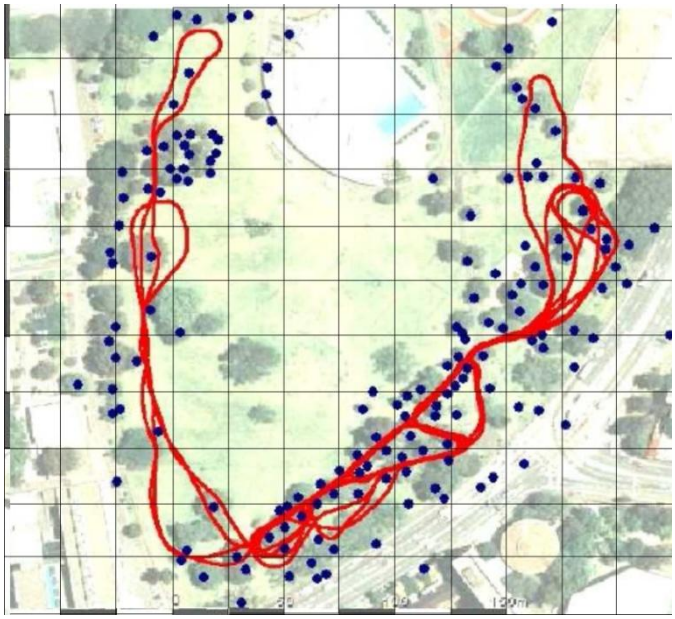


Figure 35 Victoria Park with Labeled Landmarks

In Figure 35, dots represents the extracted and labeled trees, lines represents the trajectory of robot.

In Victoria Park map, the trees are extracted and labeled by SLAM algorithm. On the other hand, in Central Park map manual extraction and labeling is required.

4.2.1.2. Manual Landmark Labeling

In contrast to Victoria Park, Central Park has not been examined in the literature for SLAM purpose. However, because of its popularity and its 3 dimensional satellite view supplied by Google Map [86], Central Park is examined as second real world map for SLAM and map merging purpose. The manual landmark labeling is the main drawback of working in uninspected areas such as Central Park. The manual landmark labeling is the process of marking the trees on the map. In this process, 3 dimensional satellite views are used as extra information about the trees. The 2 and 3 dimensional maps are displayed for illustration of the park area in Figure 36 and Figure 37.

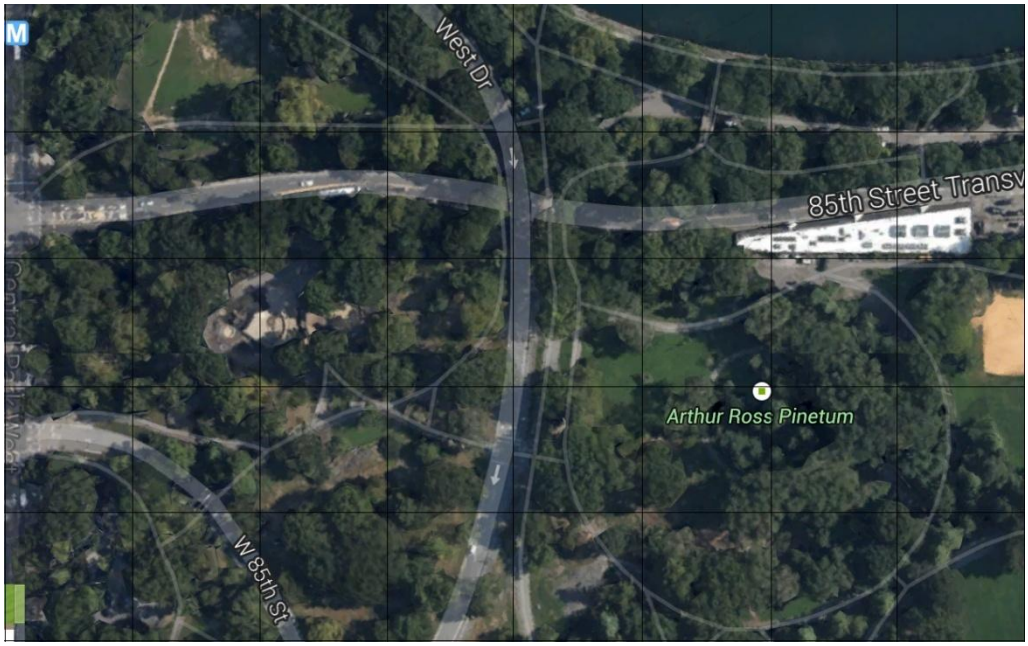


Figure 36 The 2 Dimensional Map of Central Park [86]

Figure 36, represents the partial region of Central Park around Arthur Ross Pinetum, the map is divided into 50 meters by 50 meters squares.

Figure 37 The 3 Dimensional Map of Central Park [86]

Figure 37 represents 3 dimensional view of the partial region of Central Park around Arthur Ross Pinetum. This view is used for understanding and labeling the trees in Figure 36.

The second step of manual landmark labeling is labeling the trees in the map. In this step, by using the 3 dimensional view landmarks are marked with the simple graphics painting program. The final view of map is illustrated in Figure 38.

Figure 38 The 2 Dimensional Map of Central Park with Labeled Trees

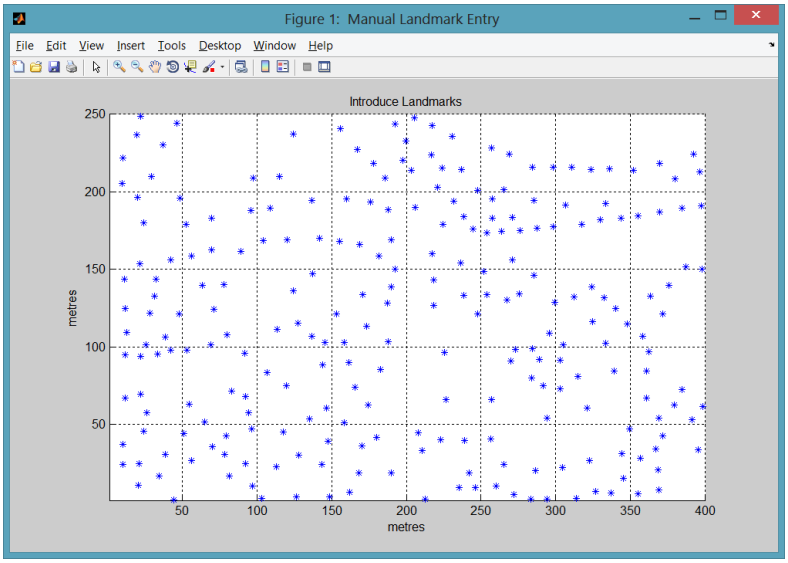


Figure 39 Manual Landmark Entered Landmarks of Central Park

In the final step of manual landmark labeling, simulator manual landmark entry option is used. By using this interface, landmarks are saved as two-dimensional

MATLAB array format. The user interface of the simulator in manual landmark entry mode is illustrated in Figure 39.

The landmarks are entered as represented in Figure 39 by using the labels in Figure 38. These landmarks are saved as "landmarks.mat" file under the related directory "Datasets\Central Park". This saving process enables the simulator for searching the directory and obtaining the related landmarks for simulation purpose as explained in the section 4.1.

4.2.2. Extraction of Different Trajectories

The trajectory of a robot consists of the waypoints' x and y coordinates. These coordinates represent the positions of iterative checkpoints on the map. While working with real world datasets, these waypoints are also supplied with experiment data. The trajectory of the robot or vehicle is given as odometer data, this data is supplied to the algorithm and next position is predicted. On the other hand, these waypoints can also be simulated by using random point generators or by entering them manually. Using random point generators can be used for performance analysis of algorithms, but they are not appropriate for realistic scenarios. Therefore, the real world data sets and manual trajectory planning are used in this study for performance tests. In this section, Victoria Park and Central Park are examined for extracting meaningful trajectories.

4.2.2.1. Obtaining Meaningful Trajectories

In Victoria Park map, the trajectories similar with Thrun's multi robot SLAM scenario is created [1]. These trajectories are the obtained by splitting the full trajectory into eight disjoint sequence for multi robot SLAM purpose. The four of these trajectories are selected for map merging, according to their overlapped regions. The overlapped region is the first requirement of calculation of relative frame transformation. Therefore, the following trajectories are selected and entered by using the manual trajectory ability of simulator, which are illustrated in Figure 40.

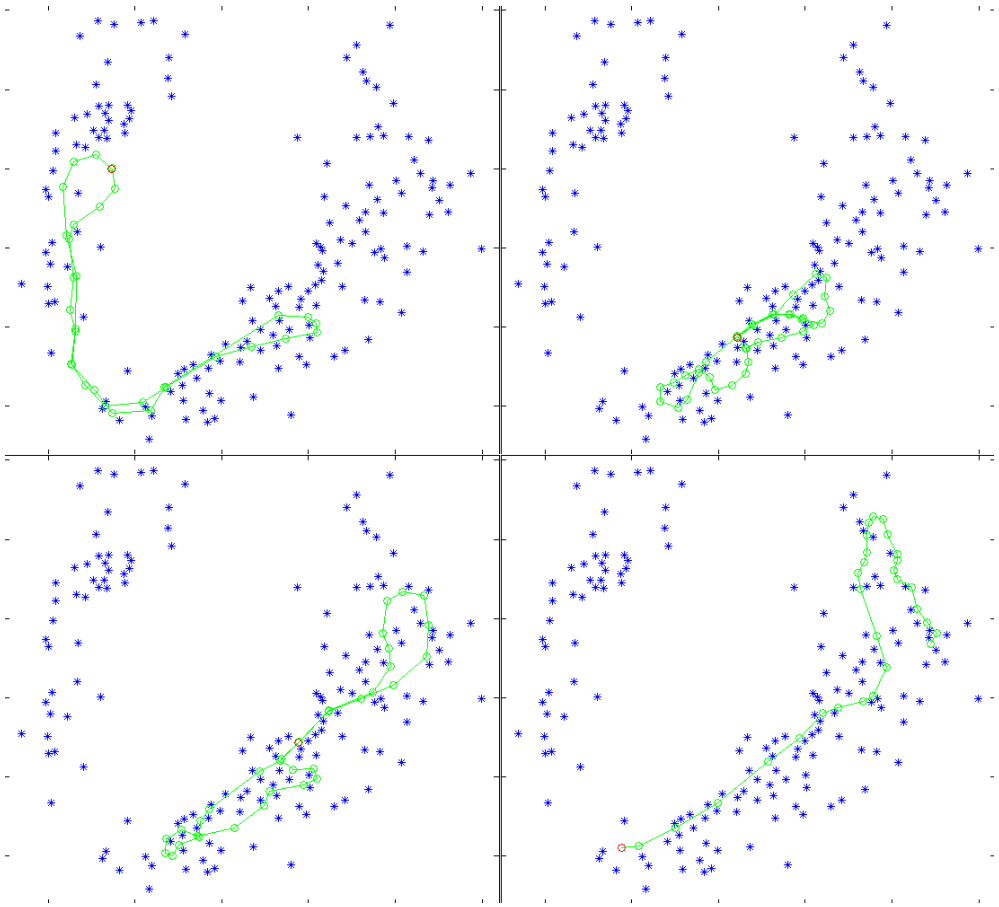


Figure 40 Victoria Park Trajectories

In contrast to Victoria Park, Central Park has no predefined trajectories. Therefore, a manual trajectory planning is required for this map.

4.2.2.2. Manual Trajectory Planning

In the Central Park map, the trajectory is not available, so the manual trajectories are created. These trajectories are planned by using the possible paths in the park. These paths can be seen in Figure 36. The other planning criterion is the overlap requirement of relative frame transformation. Therefore, paths are designed with overlapped paths. For entering these trajectories, simulator manual trajectory entering ability is used as mentioned in the section 4.1. The resulting trajectories are illustrated in Figure 41.

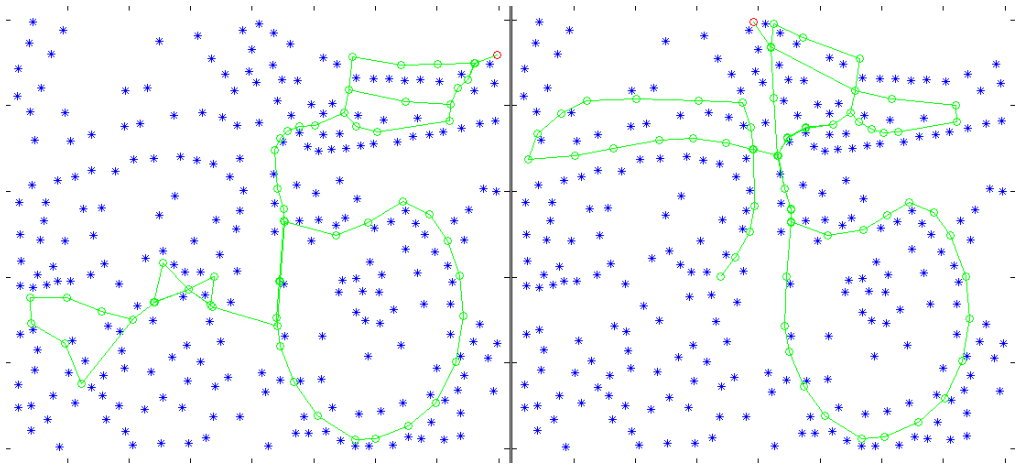


Figure 41 Central Park Trajectories

CHAPTER 5

EXPERIMENTAL RESULTS

In this chapter, the experimental procedures are presented and results are given for the selected map similarity, global map transformation and map merging algorithms together with a detailed analysis.

In this study, the Delaunay Triangulation algorithm is used for geometric map feature extraction, (explained in detail in Section 3.3.2.1.1). The extracted circumference and area from the triangles' are used to find map similarity (given in Section 3.3.2.1.2). These similar parts are the possible overlapping areas between the maps. Then RANSAC is used to search these possible areas to achieve the global map transformation, this process is explained in Section 3.3.2.2.2. The features in the overlapping regions, found by RANSAC, are the same in different maps. These matched features are merged with the Maximum Likelihood Estimator, Modified M-Estimator and the Covariance Intersection Estimator. Figure 42 shows a flow chart of the algorithms used in the study.

Firstly, computational cost analysis of implemented algorithms and their run time performances are examined, also these algorithms parameter sensitivity analysis are done. The performance experiments undertaken using the Delaunay

Triangulation, similarity metric calculation and the RANSAC algorithms are given in Section 5.1.1. The performance experiments of the Maximum Likelihood Estimator, Modified M-Estimator and Covariance Intersection Estimator are given in Section 5.1.2.1, and the sensitivity analysis of Covariance Intersection Estimator is given in Section 5.1.2.2.

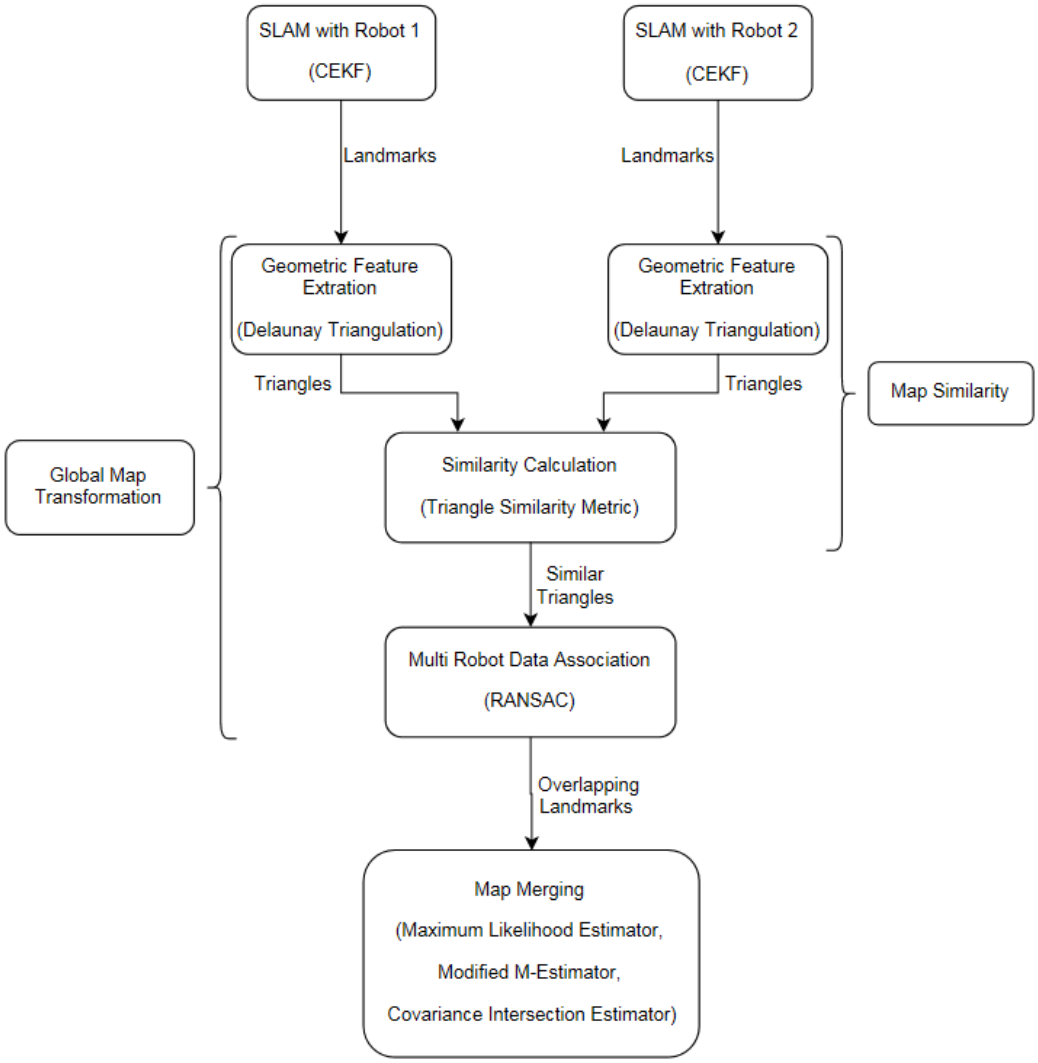


Figure 42 Multi-Robot Map-Merging Flow Diagram

The performance experiments of algorithms given above are carried out using simulated map and trajectories. The sensor bearing noise, robot velocity noise and sensor range effects on RANSAC and map similarity performance, and the effects on performance of the map-merging algorithms are presented in Section 5.2.

In Section 5.3, the performances of the algorithms, given above, are tested on real world data sets. The sensor bearing noise, robot velocity noise and sensor range effects with different trajectories on RANSAC and the map similarity performance, and the effects on performance of the Covariance Intersection and Modified M-Estimator are given.

Finally, the robustness analysis of the purposed algorithms combination, which consists of Delaunay Triangulation, RANSAC and Covariance Intersection, is given in Section 5.4.

5.1. Performance Analysis Experiments

In this section, the performance analyses of the implemented algorithms are present with a detailed examination of the behaviors of the algorithms under different conditions.

5.1.1. Global Map Transformation Algorithms

In this section, the global map transformation algorithms as explained in Section 3.4are examined in detail. Firstly, the algorithm computational costs are investigated with Monte Carlo runs with different input sets. Moreover, the algorithm parameters are tuned with respect to their computational costs, and their optimum values are selected for the following experiments.

5.1.1.1. Delaunay Triangulation

There are different implementations of the Delaunay Triangulation algorithm, which are mentioned in Section 3.3.2.1.1. This study applies the MATLAB default library implementation of this algorithm using the Computational Geometry Algorithms Library (CGAL). The computational complexity of the Delaunay Triangulation is given as $O(n * \log(n))$. In this section, computational cost of this algorithm is analyzed in detail.

The performance experiment of this algorithm requires a landmark data simulation. This simulation is achieved by using random map generator, (explained in Section 4.1.1). In this test, the landmark count is varied from 50 to 1000, to analyze the computational cost of the algorithm. The result of this test is shown in Figure 43, which was repeated 1000 times and the average value is displayed for outlier elimination.

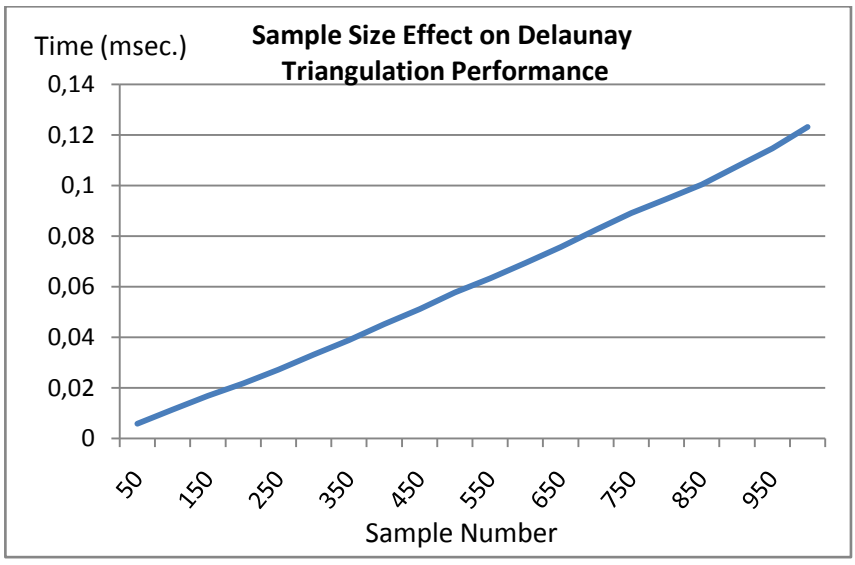


Figure 43 Sample Size Effect on Delaunay Triangulation Performance

Figure 43 shows the linear relationship between the effects of the sample size on the algorithm performance. Despite some fluctuations, the time requirement for fixed sample size being almost linear shows that the number of sample directly effects the batch calculation of the algorithm.

5.1.1.2. Similarity Metric Calculation

The similarity calculation, explained in detail in Section 3.3.2.1.2 is examined here, to better understand its computational complexity. The landmark sets of maps are transformed to the triangle sets by the Delaunay Triangulation Algorithm. These triangle sets are used as an input for the similarity metric calculation, which in turn is used for eliminating the irrelevant landmarks and finding the map similarity. By using this similarity metric, two triangle sets are compared and their most similar triangle sets generate a matched landmark pairs list. The computation cost of this metric calculation is examined with the tests given below.

5.1.1.2.1. Computational Cost

In this experiment, the similarity metric calculation is examined using different triangle sets generated using normal distributed random landmarks. Figure 44 shows the performance of similarity metric calculation with respect to sample size. This experiment is repeated 1000 times and the average value is displayed for outlier elimination.

As seen in Figure 44 the time cost of the similarity calculation is exponentially related with the sample number size. Despite this relation, the computational cost of this calculation is acceptable for 1000 landmarks, which costs about 14 msec.

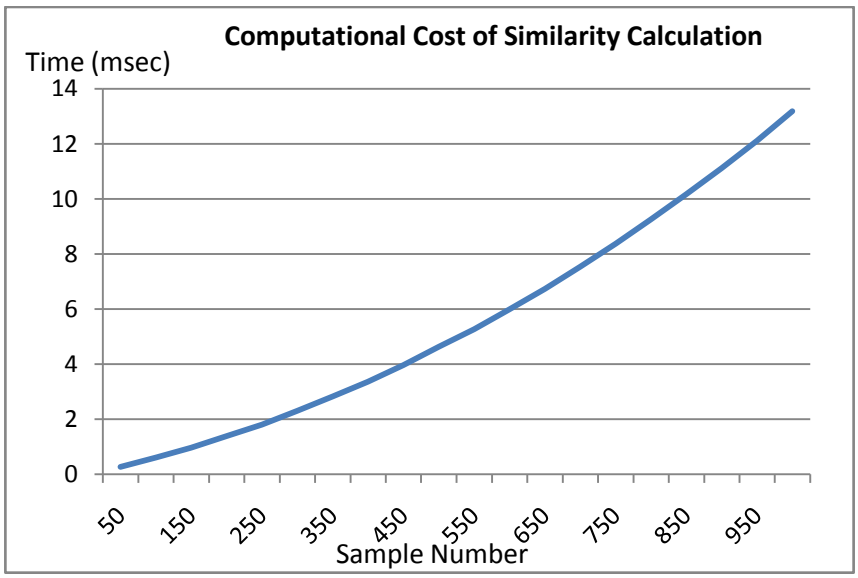


Figure 44 The Effect of Sample Size on the Similarity Calculation Performance

5.1.1.2.2. Success Rate

The similarity success rate refers to the true matched landmark pairs. The true similar landmark match ratio over all input landmark count shows the success rate. This rate is examined using different overlap ratios, which are the ratio of the landmark size of one set over another landmark size set. The overlap ratio change in the given experiment is illustrated in Figure 45.

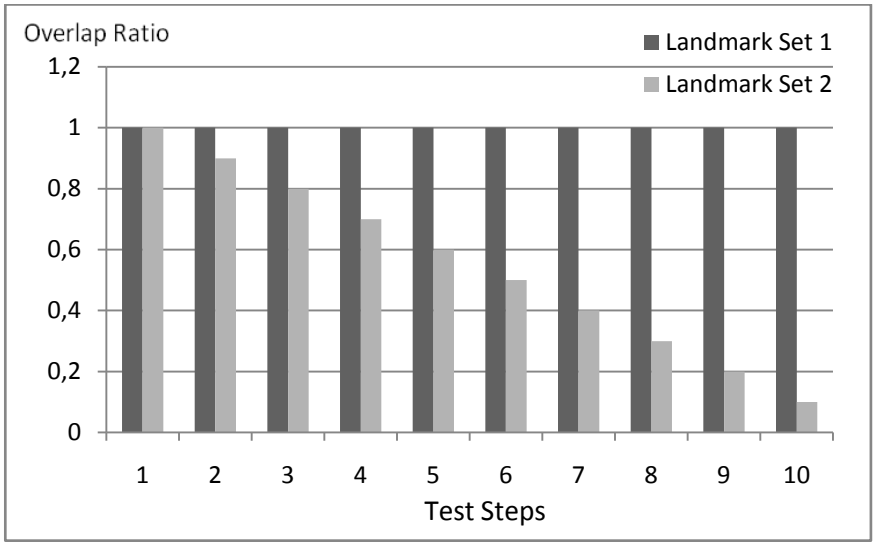


Figure 45 Illustration of Overlap Ratio Change

In Figure 45, the first darker colored bars represent the stable landmark set, which is used as the reference map, and the second bars represent the changing landmark set. The overlap ratio decrease is the decrease in the number of second landmark set. The ratio started from 100%, which is the full overlap between two landmark sets, and finished at 0.1, which means that only 10% of the landmark overlaps with the reference landmark set. The overlap criterion is the tested on the success rate of similarity calculation algorithm and the results are presented in Figure 46.

In Figure 46, effect of the ratio change on the success rate of the similarity calculation algorithm can be seen. The experiment is undertaken 100 times and average value is displayed for outlier elimination.

Although there are some fluctuations, the success rate decreases with reduces in the overlap ratio. However, in a small set of landmarks such as 100 the success rate stability even with the overlap ratio of 10% seems promising. The

experiment is also under taken for different number of landmark sets, which are given in the legend in the graph given in Figure 46. The success rate decreases with the increase in the number of landmarks. This result can be explained through the increase in the number of landmarks, the probability of finding similar pattern in the reference landmark set also increases. In fact, the results show that the overlapped region decrease affects the similarity success in negative way in large landmark sets such as 1000, which can be seen in the 0.1 overlapped region performance being below the larger overlapped region performances. However, the negative effect of the sample size and overlap region ratio does not cause low success rates in the worst-case scenario the experiment algorithm has a success rate of 99.465%. Therefore, using this similarity algorithm in the landmark elimination process before carrying out the transformation matrix search is efficient in terms of the overall performance.

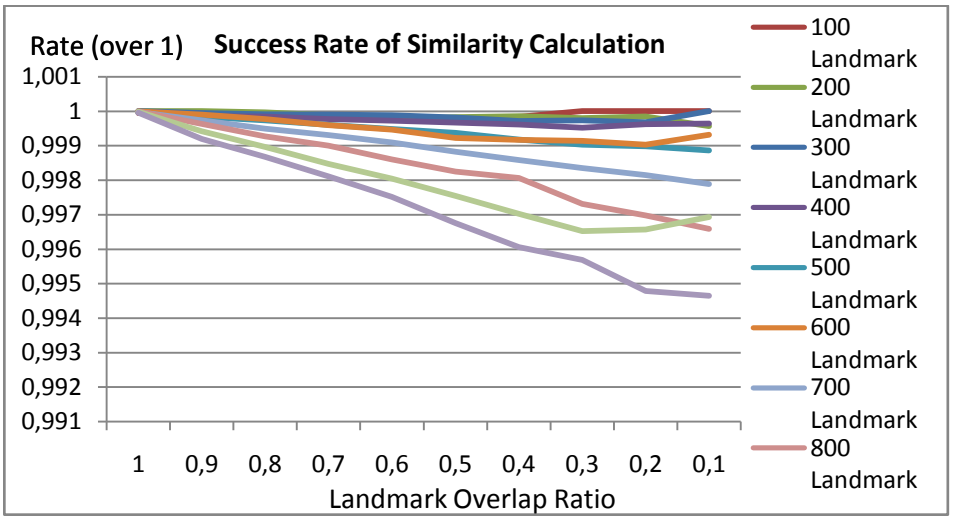


Figure 46 Effect of Overlap Ratio Change on Similarity Calculation Performance

5.1.1.3. Random Sample Consensus

This part of the study reviews the experiments carried out on the Random Sample Consensus algorithm and examines the behaviors of the algorithm under different conditions.

5.1.1.3.1. Computational Cost of Algorithm Steps

The algorithm used for transformation matrix estimation is explained in detail in Section 3.3.2.2.2. The transformation model used in the map-merging scenario is a rotation and translation matrix in two-dimensional spaces. For the initial estimate of this matrix, the algorithm takes Minimum Sample Set (MSS) randomly and derives a model hypothesis from this set. This derivation cost is minimal for a smaller sample set for the hypothesis step, however, the whole landmark set is tested with this model in test step of algorithm, which can result in high computational cost.

The first experiment is carried out to better understand the computational cost relation between the RANSAC algorithm steps. In this experiment, the hypothesis and test steps computational costs are examined with a randomly generated sample set. Figure 47 presents the results of the experiment carried out 1000 times and average value is displayed for outlier elimination.

In Figure 47, the time cost of the hypothesis step fluctuates around 0.038 msec, which shows that the time complexity of the hypothesis step is not related to the sample number. On the other hand, the test step is directly related to the sample number and it is seen that the time cost of this step increases if the sample number

increases. Another important result shown in this graph is that the hypothesis step cost starts higher than the cost of the test step. This shows that if the sample number is below 100, the computational cost of the model is larger than the test step cost. This is the result of the matrix inversion operations in the test step, which is given in equation (59).

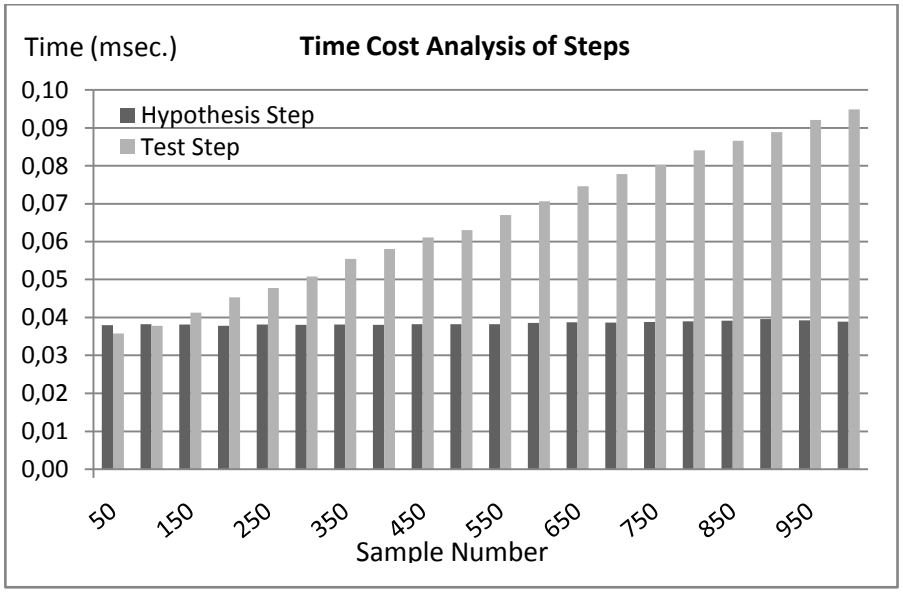


Figure 47 Time Cost Illustration of Hypothesis and Test Steps

5.1.1.3.2. Success Rate under Noisy Input

In this experiment, success rate of RANSAC with noisy input landmarks is analyzed. These landmarks are randomly distributed; also, they are disturbed by different noise levels. In this experiment, Gaussian noise with a zero mean and different sigma values is added to the randomly generated landmarks. Moreover, the different number of sample characteristics is also examined. These landmarks are given to RANSAC just to examine the effect of the noise level. The success rate of algorithm is achieved by taking the inliers ratio over the whole landmark

set. The test is done 1000 times and average value is displayed for outlier elimination, and its results are given in Figure 48,.

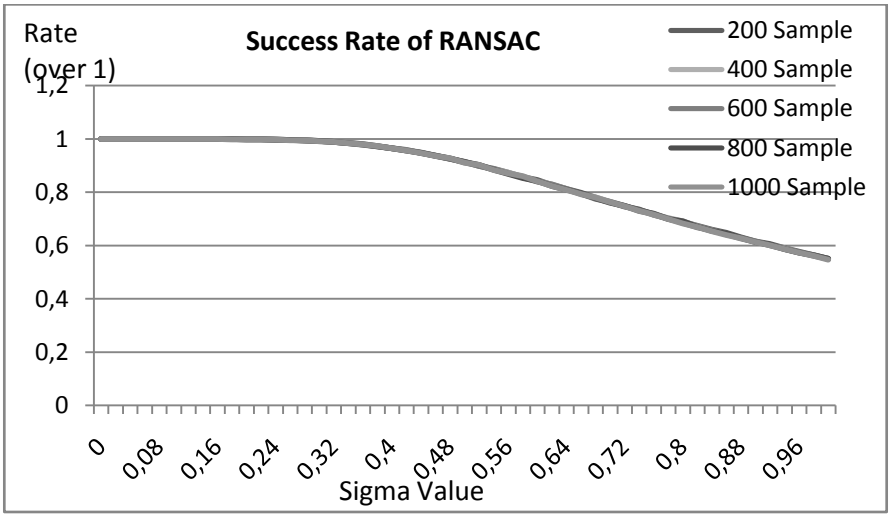


Figure 48 The Success Rate of RANSAC

Figure 48 shows that the RANSAC performance decreases with the increase in the sigma values of the landmark noises. In the graph, there is a break point at sigma value 0.3, and it can be seen that the 95% success rate is accomplished even with a 0.44 sigma value. Moreover, in the experiments a different number of sample points are tested as given in the legend of the graph in Figure 48. The sample number effect on the success rate is negligible. The distance threshold parameter of the algorithm is set to 2 meters, and the Euclidian distance is used as the distance function of algorithm. These results show that the RANSAC performance is promising under even noisy environment.

5.1.1.3.3. Success Rate under a Mismatched Noisy Input

In this experiment, the mismatched input effect on the RANSAC success rate is analyzed. The landmarks are generated by map generator in a randomly

distributed manner and they are disturbed with noise. The sigma level of the Gaussian noise, independently added to the landmarks is maintained as constant to understand the behavior of the algorithm under the different ratio levels of the correct landmark pairs. Moreover, the different number of sample characteristic is also examined. The input of this experiment consists of two landmark sets with the same mean values but disturbed with independent noises. One of these sets is selected as the reference and other set is reduced to a given ratio by elimination. The landmarks set generated by the Random Map Generator is not created by the distance limitation option, which means that the algorithm is also dealing with landmarks that are closer than the distance limit of the algorithm.

The calculation of success rate is undertaken by taking the inliers ratio over the true landmark pairs. In Figure 49, the results are given for the experiment, which was carried out 1000 times and average value is displayed after outlier elimination.

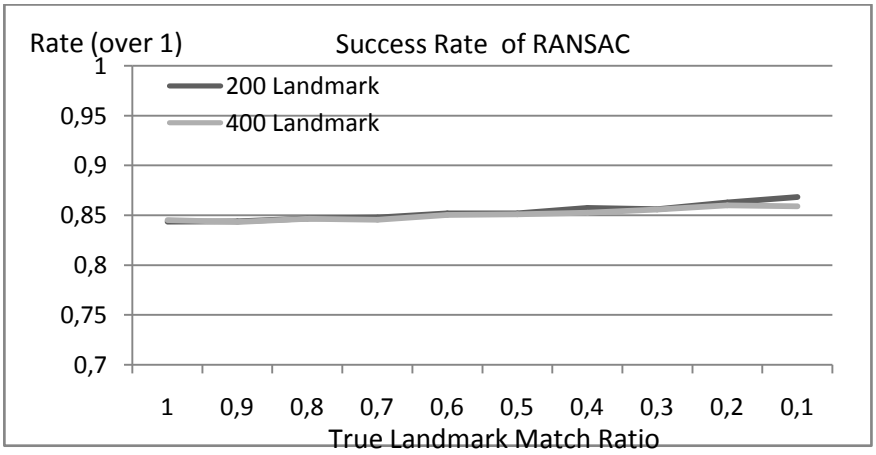


Figure 49 Success Rate of RANSAC for Different Match Ratios

Figure 49 shows that the success rate of RANSAC increases with the decrease of the true landmark pair ratio. In the graph, there are some fluctuations because of the random nature of the algorithm. The increase in the success rate can be explained by the true landmark pair affecting the overall success more than in the large set. Although the success rate increases with the match ratio decrease, this change is negligible. The increase from 0.845 to 0.87 is just 2.5% of the success rate. This experiment is tested for a different number of sample points as given in the legend of the graph in Figure 49. The larger sample variety is not tested because of the test results of the previous experimental results; Figure 48 shows that sample number effect on success rate is negligible. The distance threshold parameter of algorithm is set to 2 meters, and Euclidian distance is used as the distance function of algorithm. The sigma value of the independent landmark noise is 0.6. The selection of the sigma value is obtained from the results of the previous experiment results (Figure 48), which give a success rate of around 0.84.

These results show that, the RANSAC performance is promising even under harsher conditions such as very small true matched ratios. The true matched landmark pair ratio simulates the overlap ratio of the individual local maps of the robots. Although the success rate of the smaller match ratios are satisfactory, the calculation cost of the whole RANSAC procedure needs to be examined. Due to the random sample selection step of RANSAC, the algorithm iteration time taken in searching for the best model estimation could diverge.

5.1.1.3.4. Success Rate under Mismatched Noisy Input and RST

In this experiment, the performance of algorithm is examined under different transformation matrixes with different noise levels. The same experimental procedure given in section 5.1.1.3.3 is applied to the algorithm, but the true landmark match ratio is taken as one, because the differences in the success rates are small. The transformation matrix is also randomly generated and noise is generated with different sigma values. The calculation of the success rate is carried out by taking the inliers ratio over the undisturbed landmark subset. This experiment was repeated 1000 times and the average values after outlier elimination are given in Figure 50.

As shown in Figure 50, the RANSAC performance decreases with the increase of the sigma values of the landmark noises which is similar to the experimental results in section 5.1.1.3.3. These results show that the randomly generated transformation matrix does not affect the RANSAC performance, so even under a noisy environment with noisy transformation its performance is promising. The randomly generated transformation matrix simulates the local frame transformations of the robots' to that of the other robots e. Therefore, to solve the unknown initial position problem of the Multi-Robot problem, RANSAC could be used even in harsher environments.

In this experiment, the calculation time cost of RANSAC is examined in detail. The same experimental procedure as given in section 5.1.1.3.3 is applied. The time cost of calculation is measured before and after the whole RANSAC

procedure. This experiment was carried out 1000 times and the average value is displayed after the outlier elimination in Figure 51.

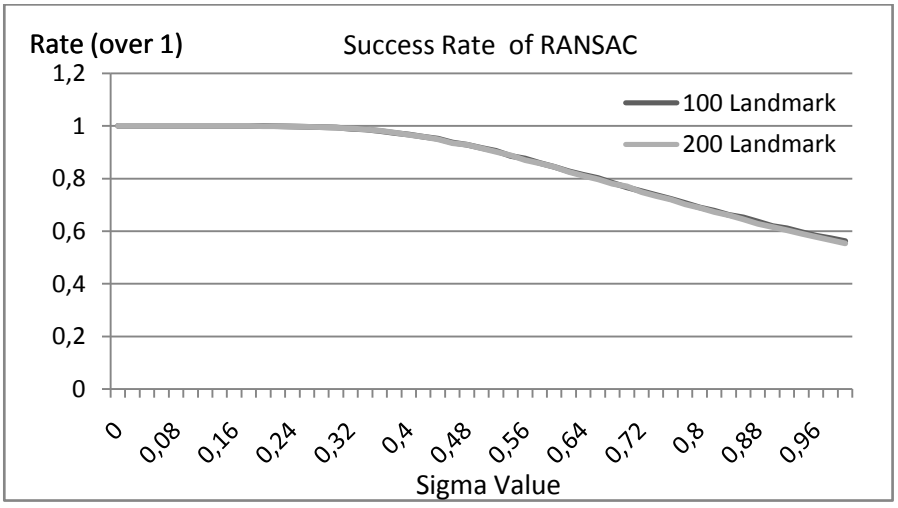


Figure 50 Success Rate of RANSAC under Different Match Ratios

5.1.1.3.5. Computational Cost under Mismatched Noisy Input

In Figure 51, the time cost of RANSAC increases with the decrease of the true landmark pair ratio. In the graph, there is a breaking point at the ratio of 0.3, which shows that the algorithm requires more iteration to find the landmark pairs in small true match ratios. Although there is an increase in the success rate shown in Figure 49, the increase in time cost is exponential. The increase of ratio from 0.3 to 0.1 results in time cost increase from 2 to 18 msec. which requires a time that is nine times longer. All the experiments tested for a different number of sample points as given in the legend of the graph. Although there is not a big difference between 200 and 400 landmarks the larger sample variety is not tested since the results of the previous experiment shown in Figure 48revealsthat the sample number effect on the success rate is negligible. The distance threshold

parameter of the algorithm is set to 2 meters, and Euclidian distance is used as the distance function of the algorithm. The sigma value of the independent landmark noises is given as 0.6. The selection of the sigma value is obtained from the previous experiment's results (shown in Figure 48), which gives a success rate of about 0.84.

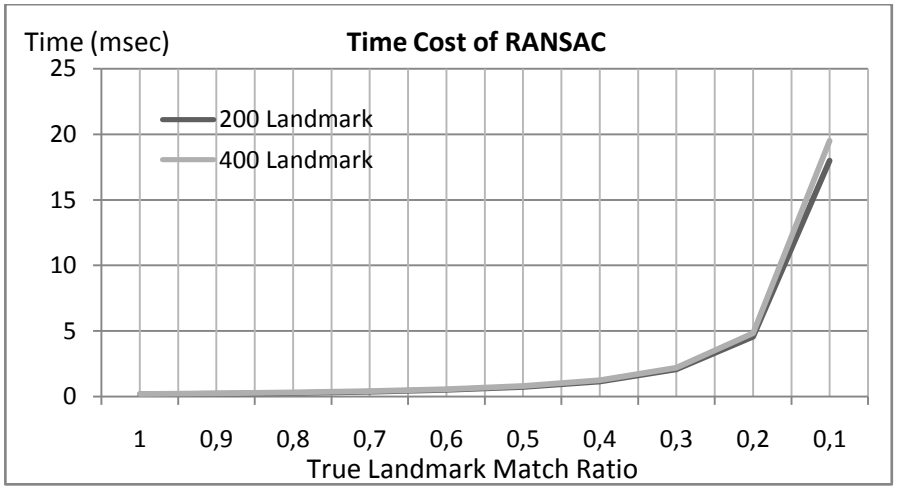


Figure 51 Computational Cost of RANSAC under Different Match Ratios

These results show that the RANSAC time requirement for harsher conditions such as very small true matched ratios is too large for real time applications. Therefore, triangulation and similarity elimination procedures are necessary for performance improvement. Since the time costs of the Delaunay Triangulation and the Similarity Calculation are independent from the true landmark match ratio, all that is the effect of sample size on these algorithms for calculations. The time requirements of the Delaunay Triangulation and the Similarity Calculation, given in Figure 43 and Figure 44, are promising. The triangulation time cost for 400 samples is around 0.04 msec. and for the similarity calculation it is around 3 msec. a total of 3.04 msec. and almost 17% of the time cost of RANSAC is under

0.1-match ratio. Therefore, the elimination procedure is crucial requirement for the real time application of RANSAC.

5.1.1.3.6. Success Rate under Different Distance Methods

This experiment examines the success rate of RANSAC under different distance methods such as the Mahalanobis and Euclidean methods. The experimental procedure given in Section 5.1.1.3.5 is applied to the algorithm. The noise sigma value, used for generated landmark positions, is also used for the Mahalanobis distance calculations. Since the experimental results in Figure 51 show that, the effect of the sample number on the success rate of RANSAC is negligible; the sample set is kept constant in this experiment. The sigma value of the noises is changed from 0.1 to 2 in the experiment, which is carried out 100 times and the average value is displayed after outlier elimination in Figure 52.

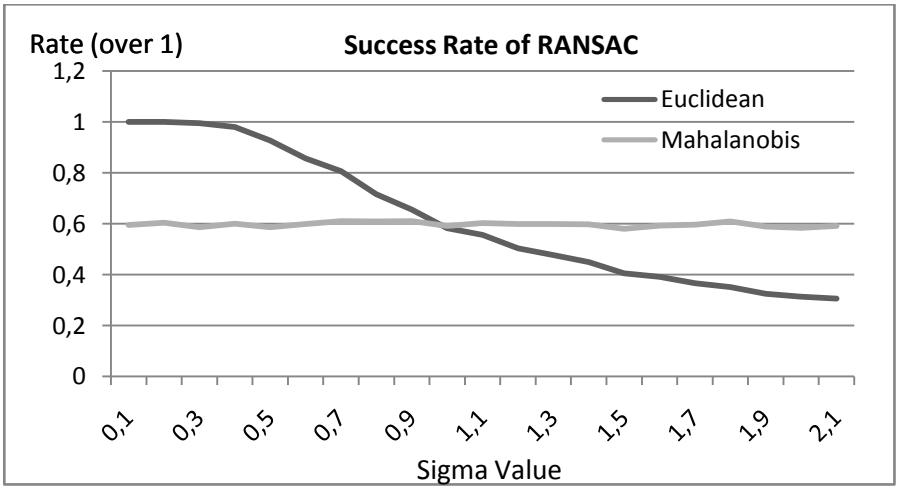


Figure 52 Success rate of RANSAC using Different Distance Calculation Methods

In Figure 52, RANSAC with Euclidean performance decreases with the increase of the sigma values of the landmark noises similar to the experiment results in Section 5.1.1.3.5. On the other hand, the RANSAC success rate using the Mahalanobis distance calculation method is almost constant around 60%. The critical point of the distance calculation method is the one sigma value. The graph in Figure 52 shows that the performance of RANSAC with the Euclidean method has a decreasing trend with an increasing sigma value. Despite successful results in the small sigma values, the Euclidean method does not give satisfactory result after the one sigma value. The main reason for the success rate decrease of the Euclidean method is the increase in the sigma value results in the landmark points too far to be handled. Therefore, if the sigma values are higher in the application using Mahalanobis distance calculation method has an advantage over the Euclidean distance calculation method. However, without examining the computational cost of these methods, this experiment is not sufficient to understand the performance of the algorithm.

5.1.1.3.7. Computational Cost under Different Distance Methods

In section 5.1.1.3.6 only the success rate of the algorithm with different distance calculation methods is examined. However, this is not sufficient to understand the real time applicability of these methods. Therefore, it is necessary to investigate the computational cost of RANSAC with different distance calculation methods. The same experimental procedure as given in Section 5.1.1.3.6 is applied to the algorithm. This experiment was carried out 1000 times, and the average value is displayed after outlier elimination and the results are given in Figure 53.

Figure 53 shows the time cost analysis of the whole RANSAC procedure. The graph shows that the cost of work with Mahalanobis distance calculation method requires more time than with the Euclidean distance calculation method. The time cost with Mahalanobis method fluctuates around 450 msec. The Euclidean method shows an increase from 4.5 msec. to 50 msec. Despite the100 times longer time cost requirement, the cost with Euclidean method is far below that of the Mahalanobis method. Therefore, with small sigma values this technique is more suited to real time applications.

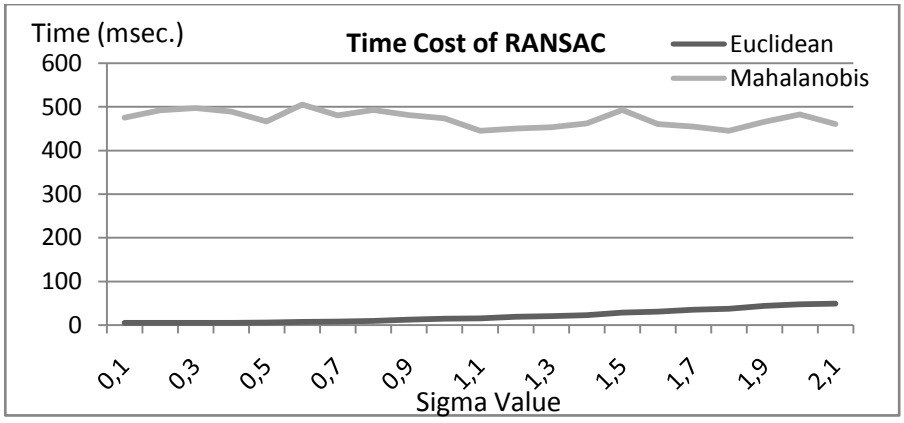


Figure 53 Computational **Cost of RANSAC under Different Distance Calculation Methods**

5.1.2. Map Merging Algorithms

In this section, the map-merging algorithms explained in Section 3.4, are examined in detail. Firstly, the algorithm performances are tuned with their parameters, and the optimum parameter set is selected for every algorithm used in the following experiments. For the evaluation of the algorithms, performance metrics are used as explained in Section 3.5, and the Monte Carlo run results are given.

5.1.2.1. Computational Cost of Algorithms

In this section, every algorithm used for map merging is analyzed with respect to their computational load; this is a very important issue in systems with a real time specification. To better understand the algorithms' computational loads; randomly generated features are merged using these algorithms. This randomness is obtained by using zero mean Gaussian distributed positions and covariance matrices. Each algorithm takes these positions and the covariance information and its computational time is recorded during this process. For stability of the performance analysis, every test is repeated 1000 times. In this test, the Covariance Intersection weight value is selected as the constant, which means no optimization is undertaken. The test results are given in Figure 54.

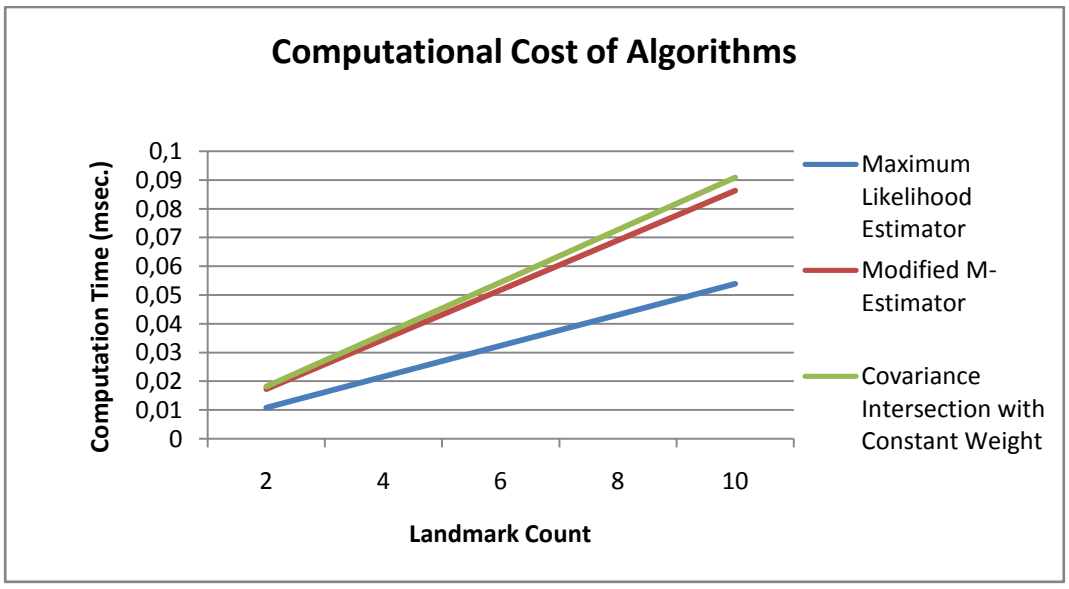


Figure 54 Computational Cost of Map Merging Algorithms

In Figure 54, Maximum Likelihood estimator's computational time is almost half that of its competitors, because it does not use the covariance information of the

landmarks. On the other hand, the Modified M-Estimator's computational time is less than that of the Covariance Intersection Estimator. The reason for this time difference is the weighting multiplication in the Covariance Intersection algorithm, given in equations (92) and (93). In fact, the computational time for both of these estimators is satisfactory for the SLAM application. For instance, the cost of merging 1000 landmarks is about 10 msec.

To understand the computational load of the Covariance Intersection algorithm, one more test is implemented using the determinant minimization of the Covariance Intersection algorithm. These results are given in Table 6 for the constant weighting parameter and optimum parameter.

Table 6 Computational Cost of Covariance Intersection Algorithm

	Time Cost for 2 landmarks	Time Cost for 10 landmarks	Time Cost for 100 landmarks
Constant weight value	0.018 msec.	0.09 msec.	1 msec.
Optimum weight value	4.3 msec.	21.5 msec.	21.5 msec.

Table 6 shows the computational time requirement of the Covariance Intersection Estimator with different weight values. It can be seen that for large landmark sets, the optimization of this algorithm requires about 20 times more time. Therefore, this algorithm sensitivity analysis should be undertaken carefully as mentioned in Section 5.1.2.2.1.

5.1.2.2. Algorithm Parameter Optimization

In this section, all the map-merging algorithms used in the experiments are analyzed with their configuration parameters. For the optimization of the algorithms, first the evaluation criterion must be clarified. The resultant feature covariance matrix's determinant and the Normalized Estimated Error Squared are also used for evaluation. Therefore, the performance of all the map-merging algorithms must be considered with these criteria.

The Maximum Likelihood estimator does not have any configurable class parameter; it just assumes the Gaussian distribution over the data and tries to minimize its error. Similarly, the Modified M-Estimator does have not any configurable parameter, so only the Covariance Intersection estimator is examined with its class parameters. The Covariance Intersection estimator has an adaptable weight parameter, which can be used for any optimization property. The most important criterion in the evaluation of the algorithms is the determinant of the resultant covariance matrix. Therefore, this parameter is changed to minimize the determinant of the resultant covariance matrix. The following experiment shows the improvement on the determinant with this optimization strategy.

5.1.2.2.1. Covariance Intersection Sensitivity Analysis

The main purpose of this experiment is to undertake a sensitivity analysis to determine the weight parameter of the Covariance Intersection estimator. For this purpose, the robots follow the same circular trajectories on the given simulated map as a test scenario, as given in Figure 55.

The scenario given in Figure 55 is implemented with two different robots, which have different sensors and characteristics. In this experiment, the first robot is identical to the second robot, but has a constant1-degree sigma bearing noise. On the other hand, the bearing sigma value of the second robot is changed from 1 to 4.5 degrees. This difference affects the estimated landmark covariance matrices, which can be seen in Figure 55. After both robots complete their trajectories, their maps are merged using the Covariance Intersection estimator. The following experimental results show the different performances of estimator under different weight values, as given in Figure 56 and Figure 57. These results are obtained from an average of 500 Monte Carlo runs.

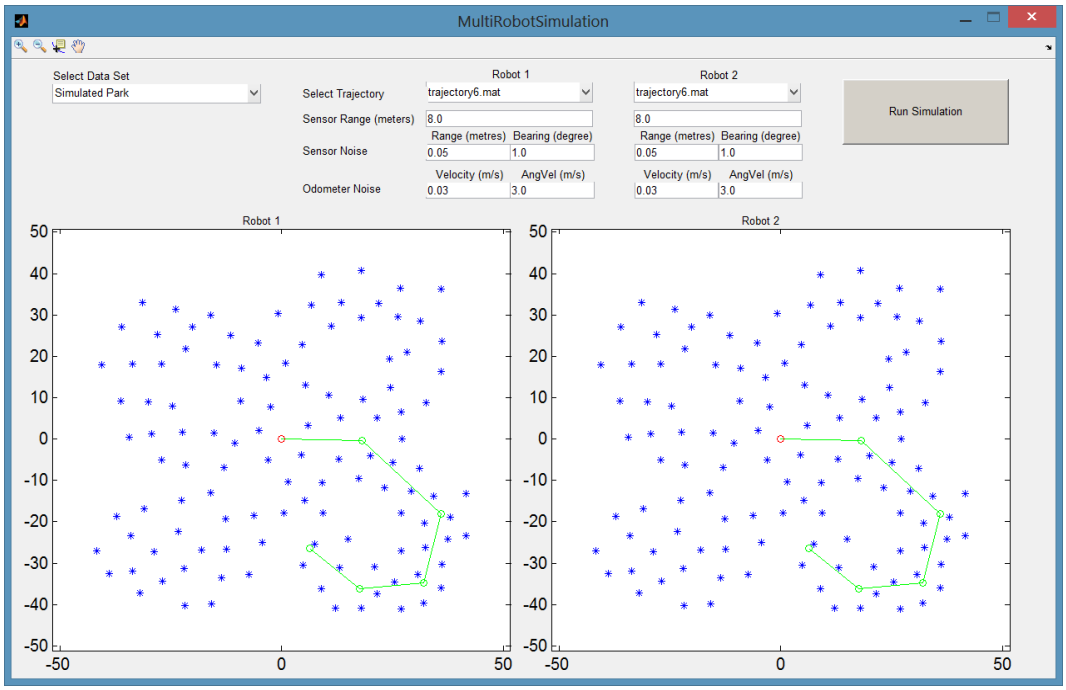


Figure 55 Simulation Scenario of Sensitivity Analysis of Covariance Intersection

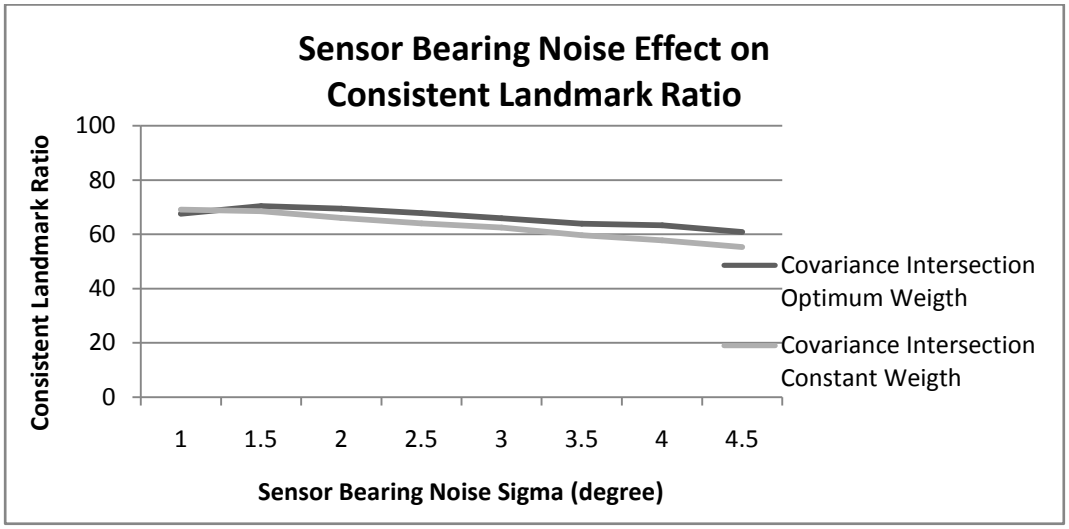


Figure 56 Sensor Bearing Noise Effect on the Consistent Landmark Number

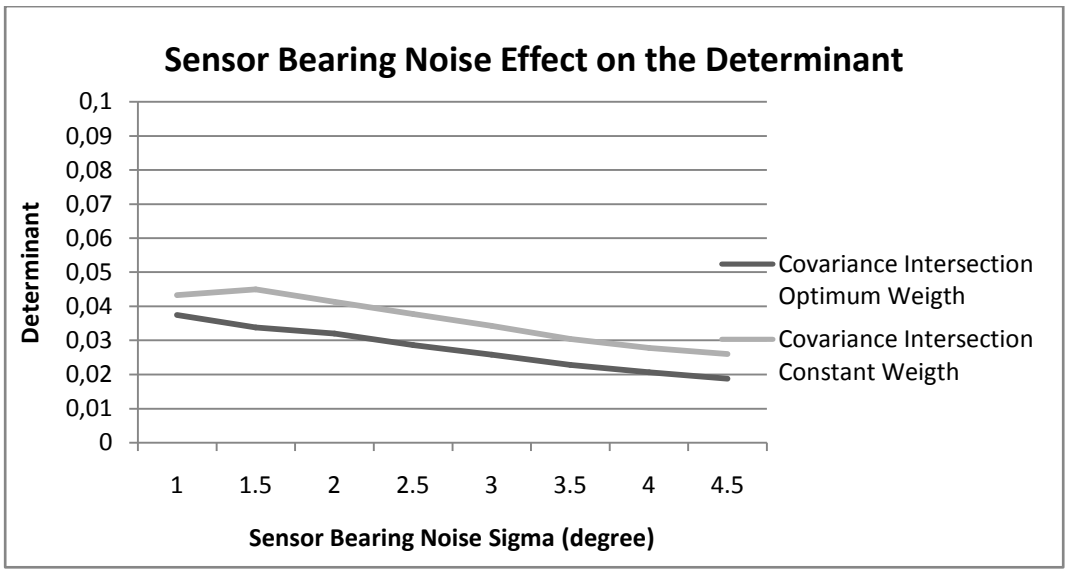


Figure 57 Sensor Bearing Noise Effect on Determinant

In Figure 56, it can be seen that the optimum weight value gives almost same results as the constant weight value. This experiment is not sufficient to evaluate different parameter performances. Therefore, for this algorithm's weight parameter selection one more evaluation criterion is required that of the area of covariance ellipses. In Figure 57, shows that the Covariance Intersection estimator

with an optimum weight parameter always has the lowest value, as expected. These experimental results clearly indicate the trade of between minimum determinant and consistency. In fact, there is no perfect parameter configuration for this algorithm therefore; the results of the experiment given in Table 6 are the main criterion for parameter selection. Since there is a high time cost in the optimum weight calculation, a constant weight is used in the following experiments. There is no preferred robot map, so 0.5 is used for the equal weighting between the robots' maps.

5.2. Simulated Map and Trajectory Experiments

This section examines the performance of the map merging algorithms using different simulated maps and trajectories. All the algorithms are used with their optimum parameter sets, as calculated in Section 5.1. For easy understanding, merged landmarks covariance matrices are plotted for 2.448 sigma region in Figure 58.

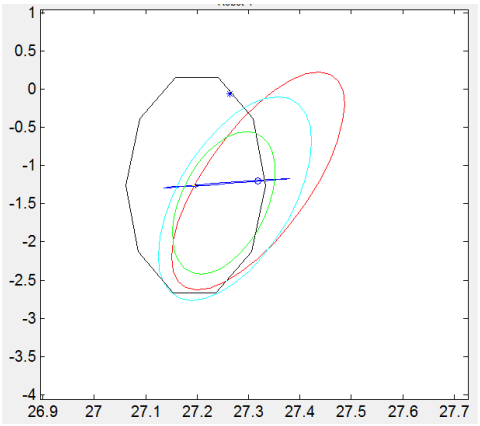


Figure 58 Covariance Estimations of Map Merging Algorithms

In Figure 58, the black and red ellipses represents the robots' landmark estimates, the blue, green, cyan ellipses represents Maximum Likelihood, Modified M-Estimator and Covariance Intersection estimations, respectively.

5.2.1. Sensor Bearing Noise Effect on Estimator Performance

In this experiment, the robots' sigma value sensor bearing noise effect on estimator performance is analyzed to better understand the estimator performances under different noise levels. Different estimator classes are examined with a simulated map with a circular trajectory as shown in Figure 59.

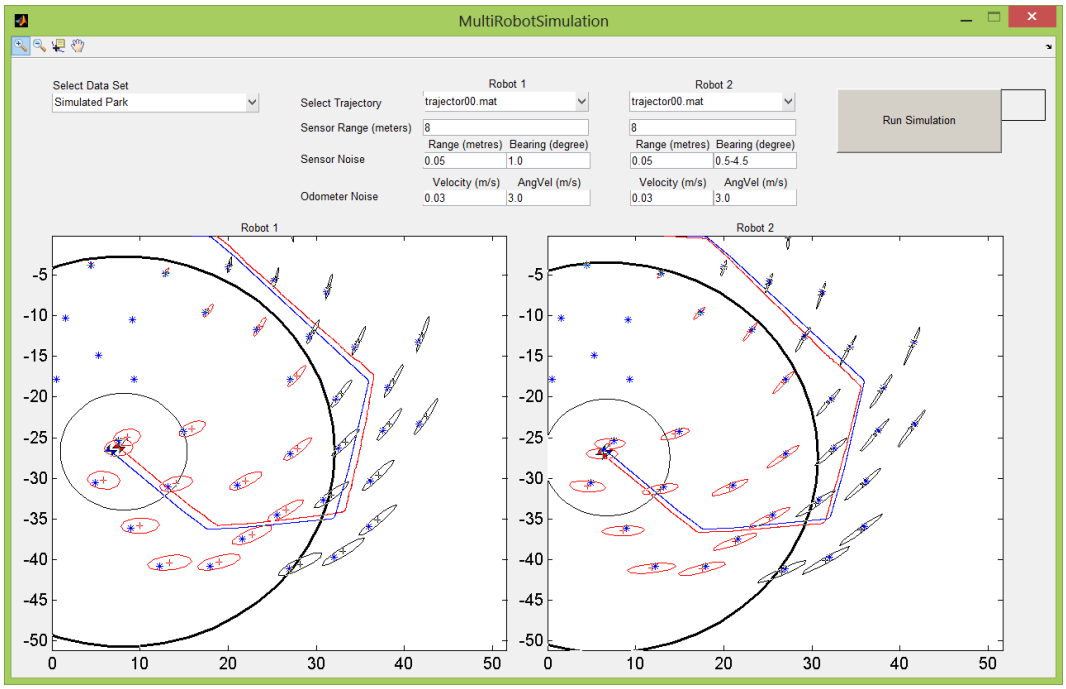


Figure 59 Simulated Map and Robot Trajectories

During this experiment, the first robot sigma values are kept constant, while second robot's bearing sigma value is changed from 0.5 to 4.5 degrees. The similarity metric success ratio and the RANSAC success ratio are given in the Sensor Bearing Noise Effect on RANSAC Performanceas shown in Figure 60.

The map merging algorithms results are given in Figure 61 and Figure 62, they are obtained with an average of 500 Monte Carlo runs.

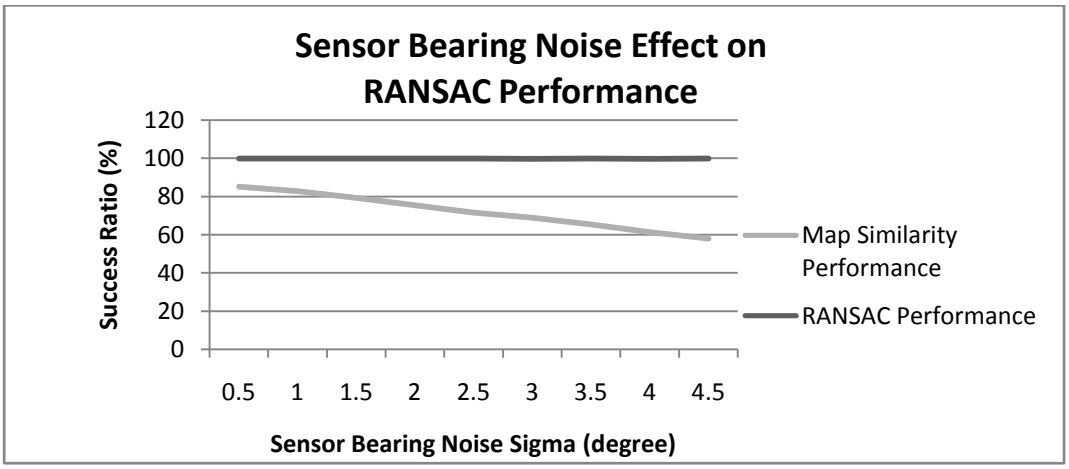


Figure 60 Sensor Bearing Noise Effect on RANSAC Performance

In Figure 60, it can be clearly seen that the performance of map similarity decreases with the increase in sigma value, which is an expected result. Since the trajectories of robots are the same, RANSAC performance is always successful for the map similarity performance above 50%.

In Figure 61, the Sensor bearing noise effect on consistent landmark ratio is given for different estimators and it can be observed that the consistency of the Maximum Likelihood estimator is not satisfactory for every sigma value. On the other hand, the Modified M-Estimator gives satisfactory results, but Covariance Intersection estimator gives the best results. In fact, this experiment also shows that the different sigma values of sensors do not affect the consistency, because the landmark counts are kept almost constant. Although, these results show that maximum likelihood does not compete with the other evaluation criterion, which

is the determinant of the resultant covariance matrix that is tested for a better evaluation of all the algorithms.

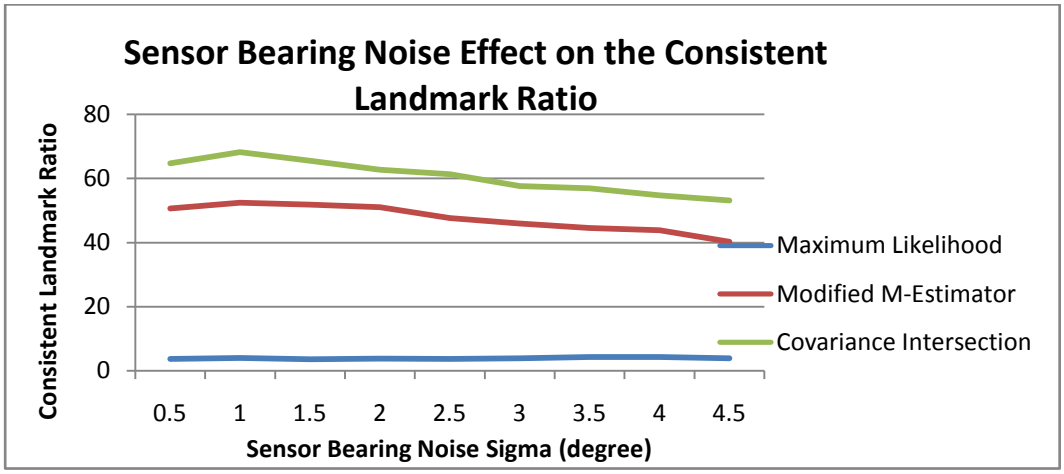


Figure 61 Sensor Bearing Noise Effect on the Consistent Landmark Number

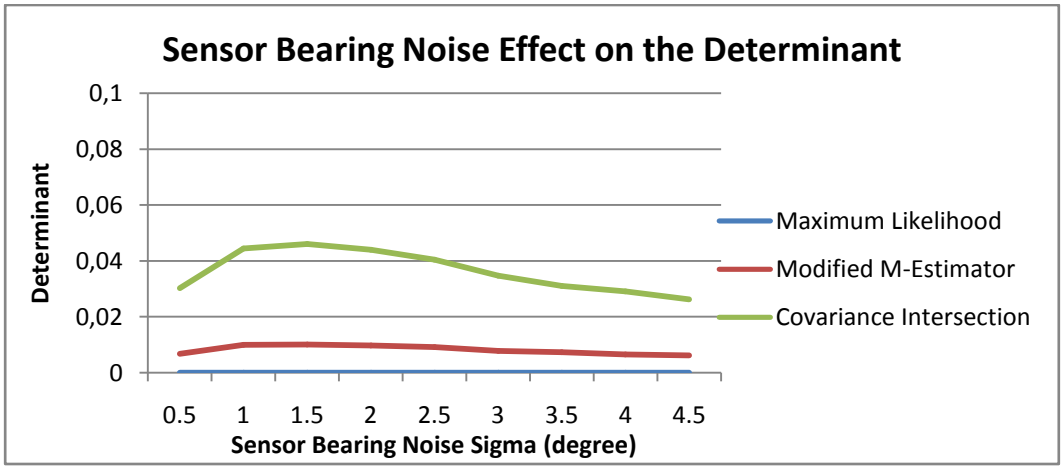


Figure 62 Sensor Bearing Noise Effect on the Determinant

The sensor bearing noise effect on resultant covariance determinant value is presented in Figure 62 and it is seen that Maximum Likelihood estimator always has the lowest value, but it is meaningless to use inconsistent values covariance determinant for comparison. Therefore, the other algorithm results are used for the evaluation. Although the Modified M-Estimator has a lower determinant value

than the Covariance Intersection estimator, its consistency decreases from 50% to 40%as seen in Figure 61. However, the Covariance Intersection estimator consistent landmark ratio decreases from 65% to 55%, and is always higher than the Modified M-Estimator's ratio.

In this experiment, the Covariance Intersection estimator gives more consistent results than its competitors. However, its determinant values for the estimated covariance matrix is higher than its competitors. As stated above, it is meaningless to use inconsistent values of the covariance determinant for comparison. Therefore, the conclusion is that the Covariance Intersection algorithm performance is satisfactory and better than its competitors.

5.2.2. Robot Velocity Effect on Estimator Performance

In this experiment, the robot's velocity noise effect on estimator performance is analyzed for a better understanding of the estimator performances under different noise levels. To achieve this different estimator classes are examined with the same map and trajectory as in the experiment in Section 5.2.1.

Rather than changing sensor bearing noise level, the robot's velocity noise level is changed from 0.05 to 0.45 meters while the first robot's velocity sigma level is kept constant at 0.03 meters. The similarity metric success ratio and RANSAC success ratio are given in Figure 63. These experiment results are obtained from an average of 500 Monte Carlo runs and given in the Figure 64.

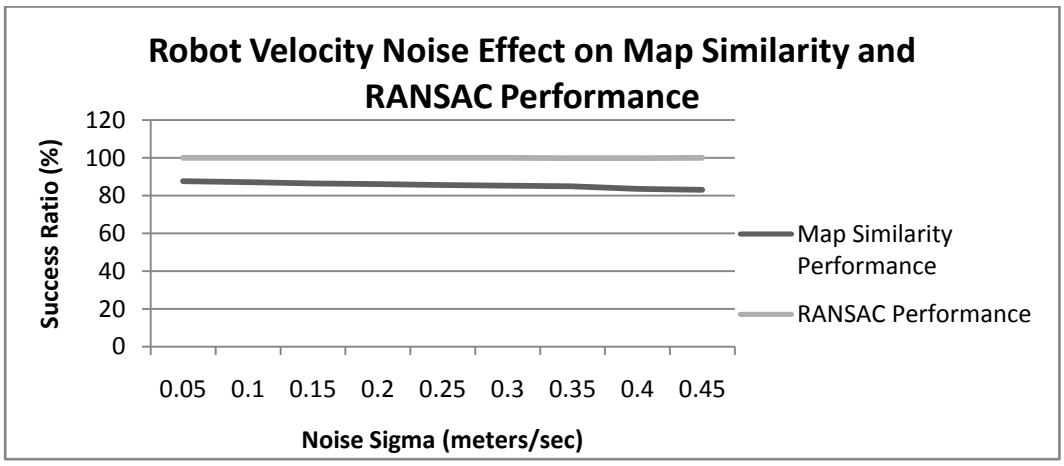


Figure 63 Robot Velocity Noise Effect on RANSAC Performance

This experiment gives similar results to the experiment in Section 5.2.1, which shows the consistent performance of the estimators under different noise level of robot's velocity. Moreover, it is concluded that the Maximum Likelihood is not applicable in distributed systems, because its results are far below than its competitors. Therefore, this estimator is omitted from the following experiments.

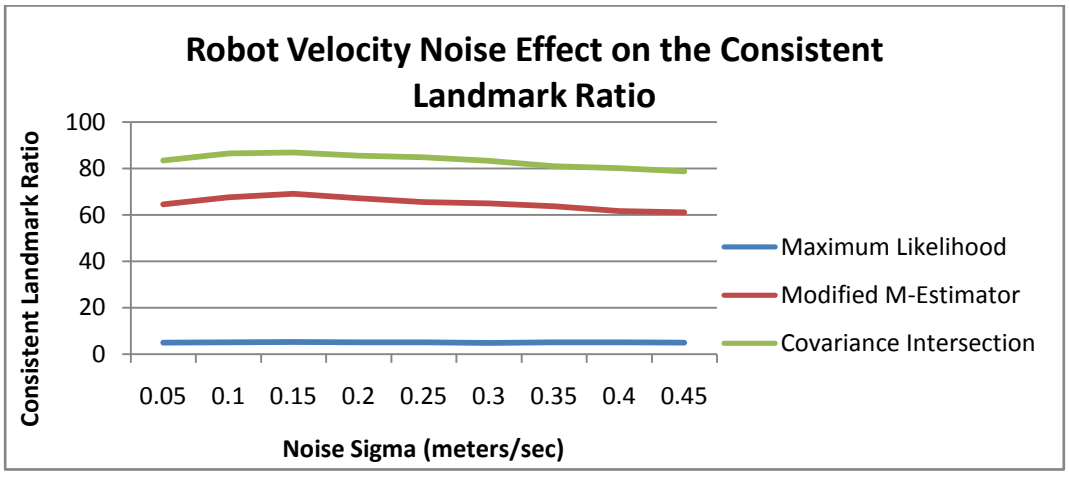


Figure 64 Robot Velocity Noise Effect on the number of Consistent Landmarks

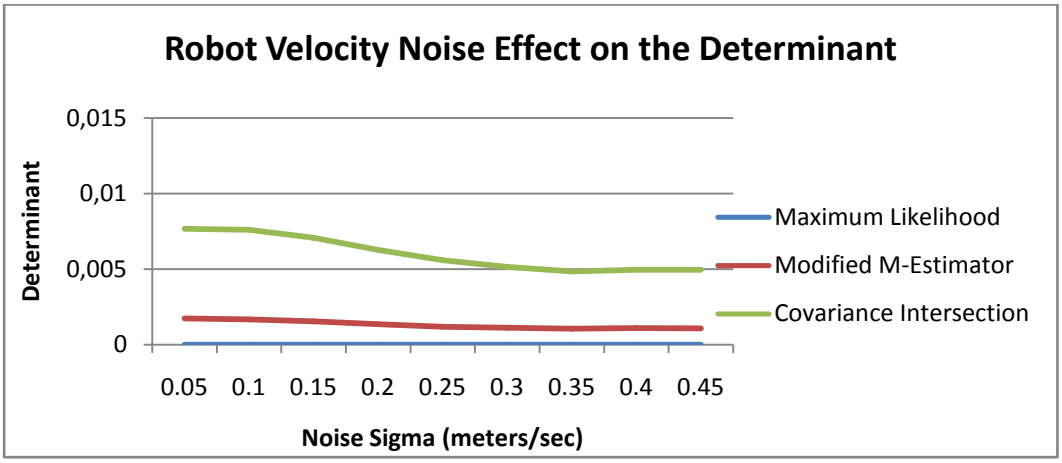


Figure 65 Robot Velocity Noise Effect on the Determinant

5.2.3. Trajectory and Sensor Bearing Effect on Map Merging Performance

In this experiment, the effect of the trajectory and sensor bearing noise on estimator performance is examined for a better understanding of the estimator performances under different noise levels with different trajectories. Different estimator classes are examined with simulated map with circular trajectory, which is given in Figure 66.

In this experiment, the robots' sigma value of bearing noise effect on the estimators' performance is examined. During this experiment, the robot sigma values are kept constant, while the second robot's bearing sigma value is changed from 0.5 to 4.5 degrees. The similarity metric success ratio and RANSAC success ratio are given in Figure 67. These results of the experiment of the map merging algorithms, obtained with average of 500 Monte Carlo runs, are given in Figure 68 and Figure 69.

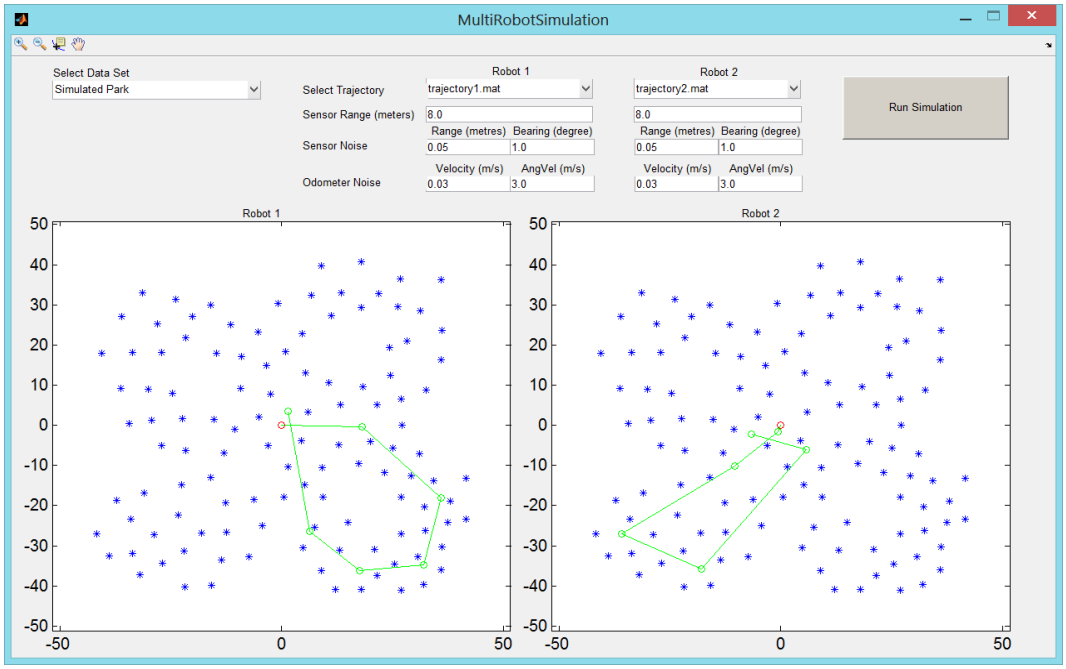


Figure 66 Simulated Map and Trajectory Illustration

In Figure 66, the overlapping area is around 5% of the total explored area, which makes it very difficult to find the global transformation matrix.

Figure 67 clearly shows that the performance of map similarity and RANSAC decreases with the increase in the sigma value, which is an expected result. Furthermore, the similar characteristic can also be seen in the graph, this indicates that map similarity and RANSAC are highly correlated with each other. It can also be seen that RANSAC performance is satisfactory even with a very low map similarity performance. For instance, RANSAC success ratio is 60% for 40% success ratio of map similarity.

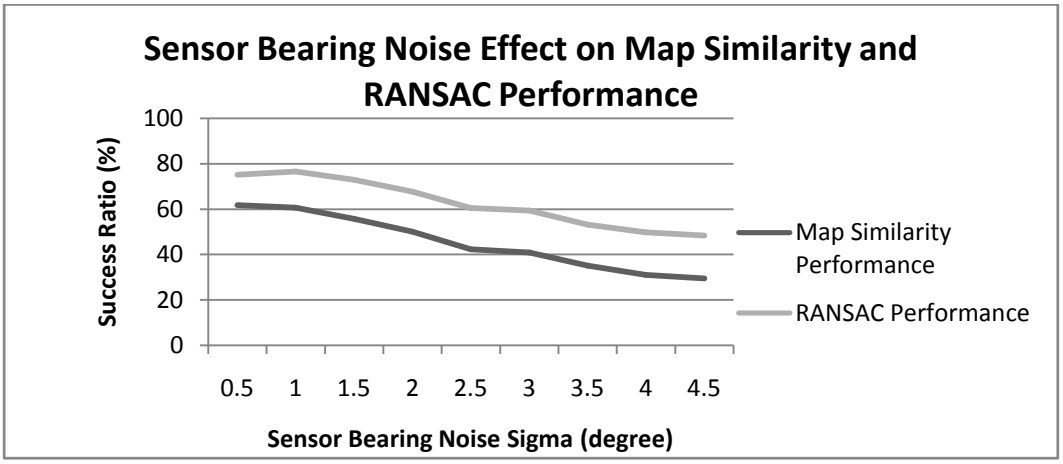


Figure 67 Robot Velocity Noise Effect on Map Similarity and RANSAC Performance

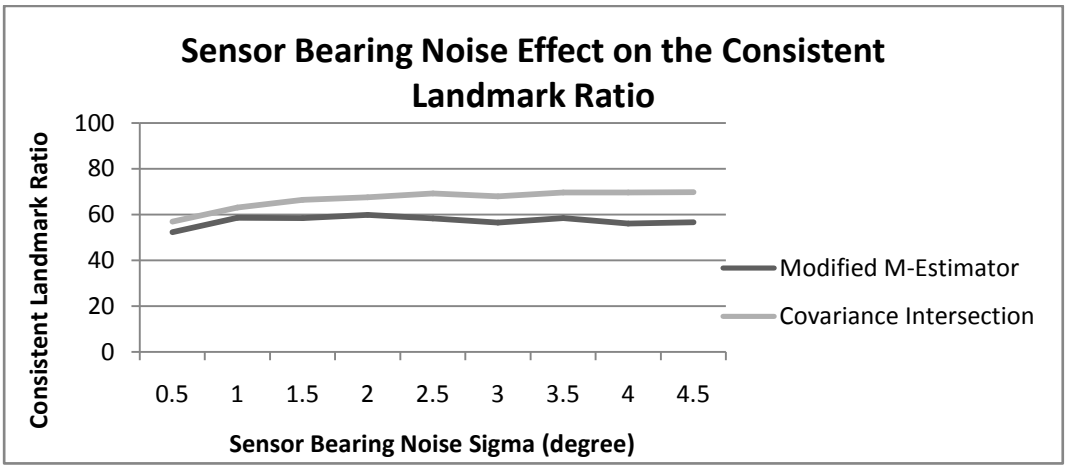


Figure 68 Sensor Bearing Noise Effect on the Consistent Landmark Ratio

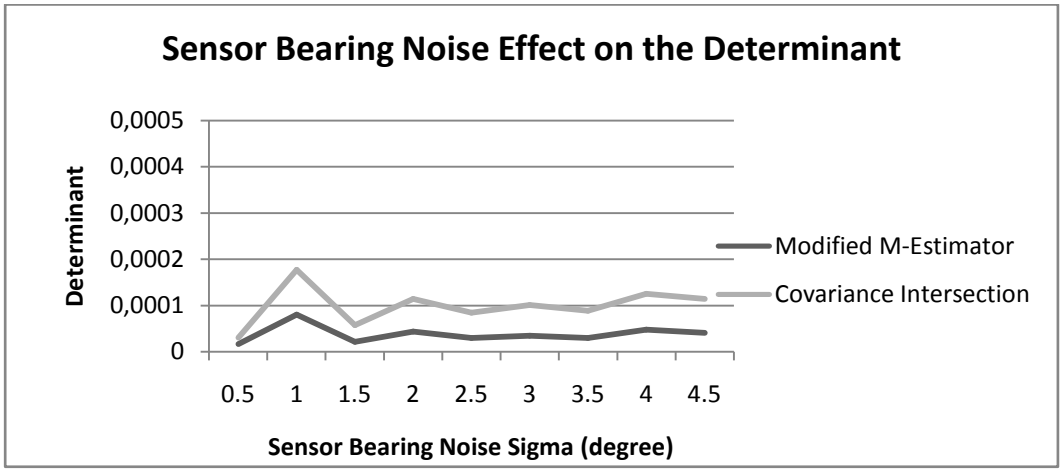


Figure 69 Sensor Bearing Noise Effect on the Determinant

In this experiment, the trajectory effect on estimators' performances is also tested under different sensor noise. These results of the experiment are given in Figure 68 and Figure 69 for the Covariance Intersection and the Modified M-Estimator. In fact, similar results are obtained from the previous experiments, this shows that the trajectory change does not affect the map merging algorithms' performances.

5.2.4. Trajectory and Sensor Range Effect on Map Merging Performance

In this experiment, the effect of the trajectory and sensor range on the estimator performance is analyzed to better understand the estimator performances under different noise levels with different trajectories. The different estimator classes are examined using a simulated map with circular trajectory as shown in Figure 70.

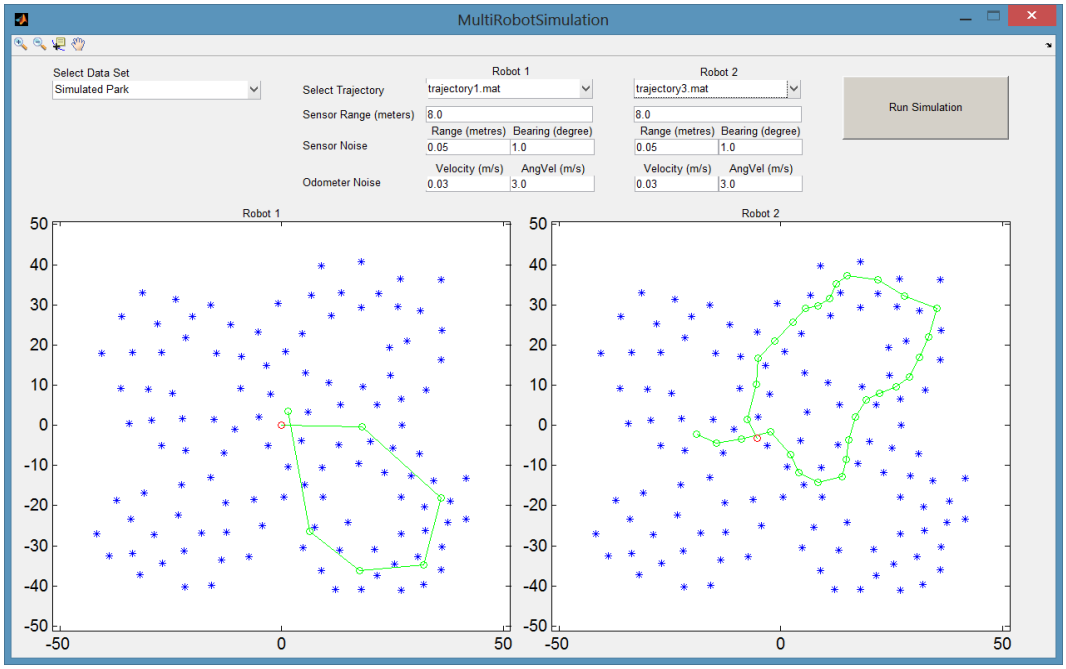


Figure 70 Simulated Map and Trajectory Illustration

Figure 70 shows an overlapping area of about 5% of the total explored area, which makes it very difficult to find the global transformation matrix. Under these limited conditions obtained with average of 500 Monte Carlo runs, the map similarity and RANSAC success performance is given in Figure 71, which are.

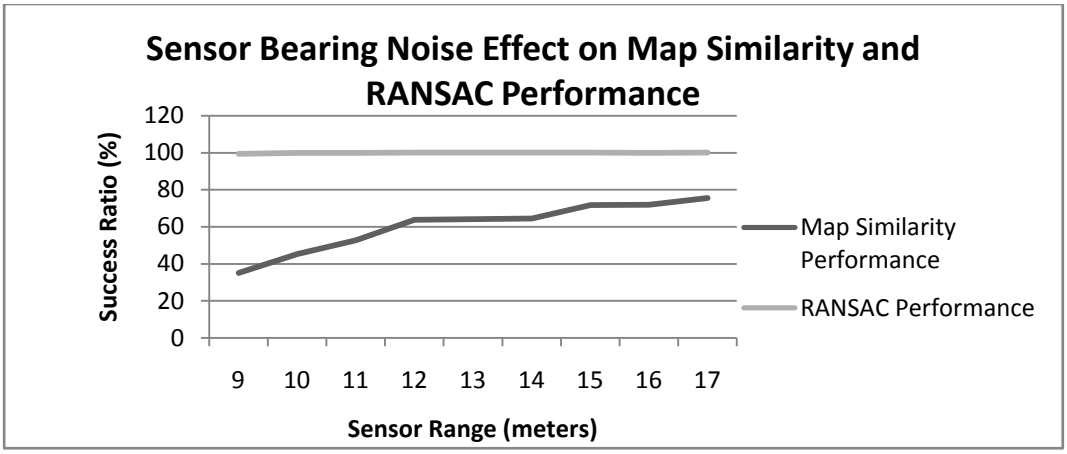


Figure 71 Sensor Range Effect on Map Similarity and RANSAC Performance

In Figure 71, shows that the performance of map similarity and RANSAC is increases with the increase in the sensor range, which is an expected result. A similar characteristic can also be seen in the graph, which shows that they are highly correlated. It is also seen that, even with very low map similarity performance, the RANSAC performance is satisfactory, at around 100% at sensor range 9 with 40% map similarity success.

The trajectory effect is tested under different sensor ranges, by changing the sensor range of the second robot from 9 to 12. The results of the experiment are given in Figure 72 and Figure 73for the Covariance Intersection and Modified M-Estimator.

In fact, similar results were obtained from the previous experiments, but consistency decreases with the increase in the sensor range. The second robot does not close its loop, when the overlapping exists. Due to the robot's position errors erroneous landmark estimates are created. Therefore, consistency decreases with the increase in sensor range. Moreover, the increase in the sensor range results in an increase in the overlapping area. However, this area is unstable and the landmarks in this area are erroneous because of the robot's cumulative position errors. As a result, this experiment shows that the increase in the sensor range can result in a decrease in consistency if the overlapping area is not sufficiently consistent.

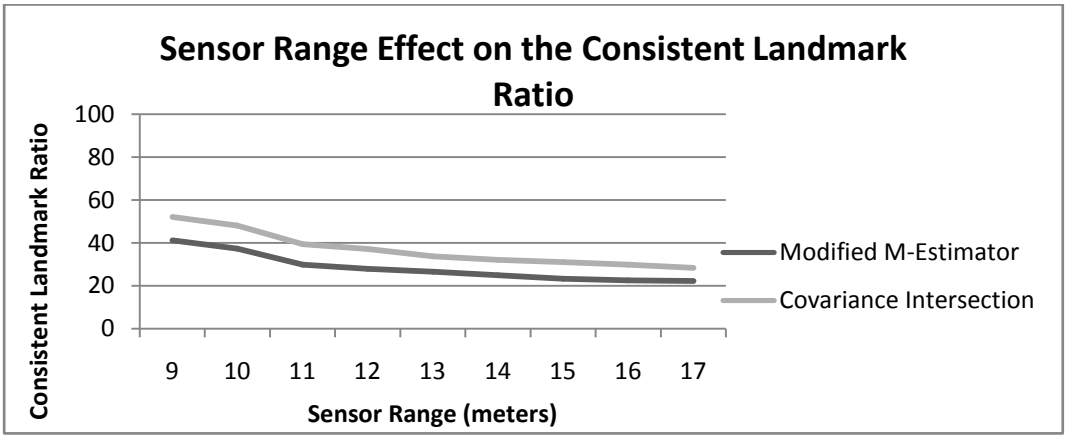


Figure 72 Sensor Range Effect on the Consistent Landmark Ratio

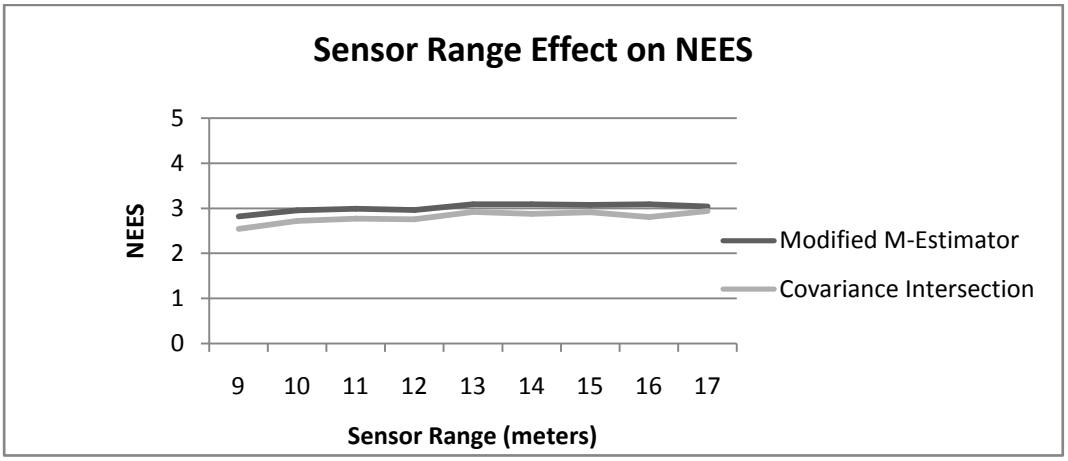


Figure 73 Sensor Range Effect on the Determinant

5.3. Real World Map and Trajectory Experiments

In this section, map merging algorithm performances are examined using real world maps and trajectories. All the algorithms are used with their optimum parameter sets as calculated in section 5.1.

5.3.1. Sensor Bearing Noise Effect on Map Merging Performance

This section presents the examination of the map-merging performance using different trajectories from the Victoria Park dataset. This dataset and the

extraction technique are explained in detail in Sections 4.2.1.1 and 4.2.2.1, and their extracted maps are given in Figure 74.

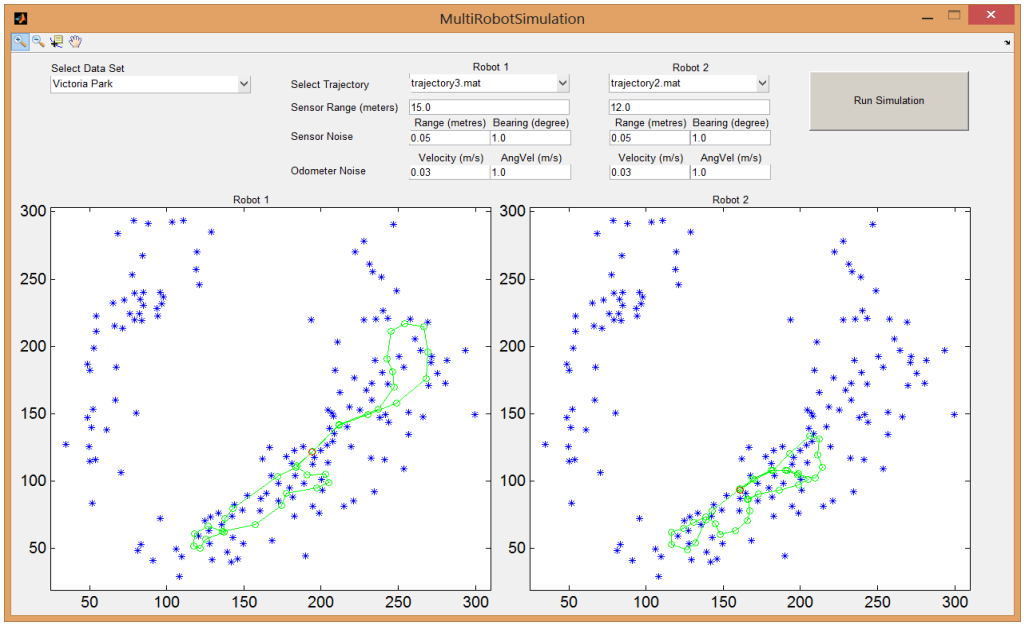


Figure 74 The Victoria Park Extracted Map and Trajectory Illustration

The robots' trajectories with their extracted map are given in Figure 74. This experiment is carried out by changing second robot's sensor sigma value of bearing noise from 0.5to 4.5. Based on these conditions, map similarity and the RANSAC success performance obtained with an average of 50 Monte Carlo runs is given in Figure 75.

Figure 75 clearly shows that the success ratios of the performance of map similarity decreases from 90% to 70% and RANSAC is stable at 100%. The stable and perfect performance of RANSAC is an expected result of successful map similarity performance. Therefore, these results are consistent with the results of the simulated data set. The successful performance of map similarity algorithm is

an expected result of closed loops in the trajectories. However, it can be seen that the map similarity performance is decreasing due to the non-deterministic SLAM outputs.

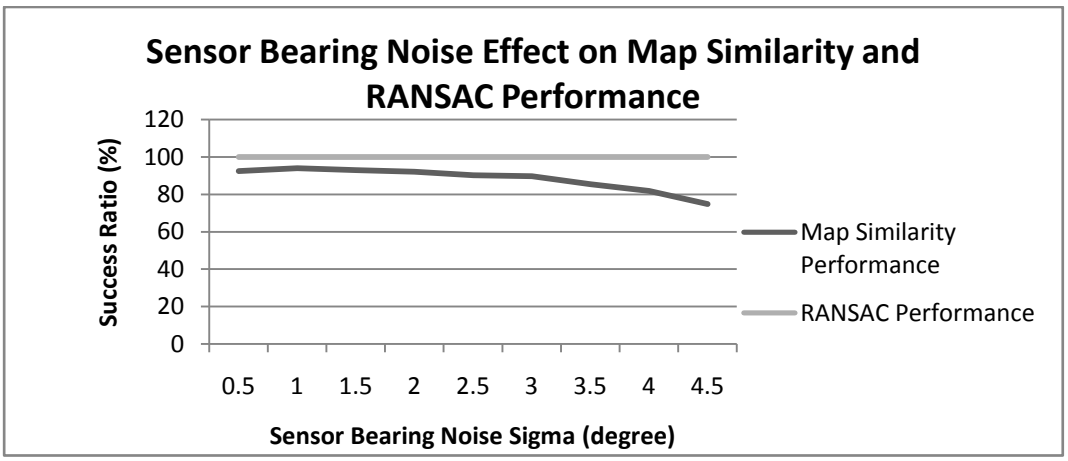


Figure 75 Sensor Bearing Noise Effect on Map Similarity and RANSAC Performance

The results of this experiment concerning the Covariance Intersection and Modified M-Estimator are similar to those observed in the previous experiments. Therefore, the consistent landmark ratio and determinant graphics are not included in this section. To better understand these methods' performances, their unfiltered results can also be examined. In fact, the consistency test does not bias their errors, but the determinant without its relationship with consistency does not give clear ranking results. Therefore, the results of Normalized Estimation Error Squared (NEES), which is unfiltered output of the estimators, explained in detail in Section 3.5, are used for better ranking and given in Figure 76.

Despite some fluctuations, Figure 76 shows that there are results that are consistent with the previous simulated environment experiment. This experiment

clearly demonstrates that the NEES of Covariance Intersection algorithm remains below the NEES of Modified M-Estimator.

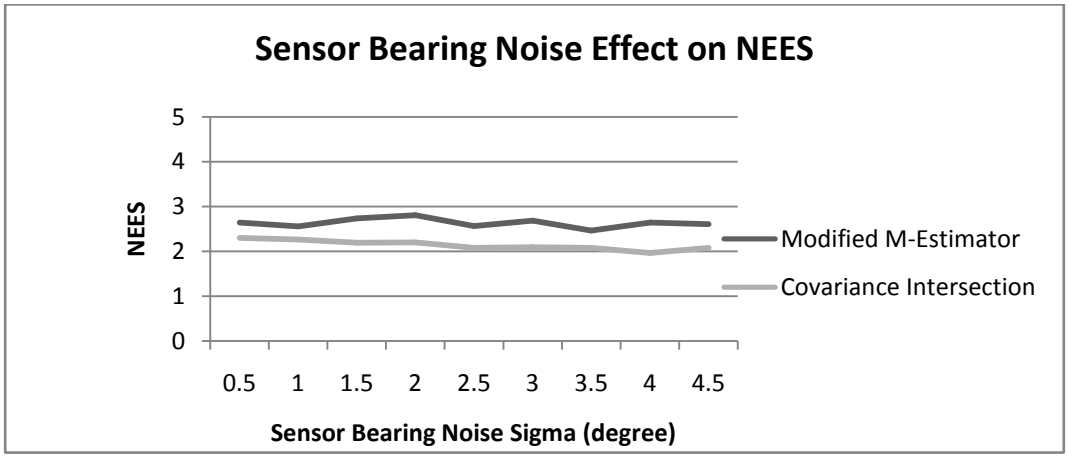


Figure 76 Sensor Bearing Noise Effect on Consistent Landmark Ratio

5.3.2. Robot Velocity Noise Effect on Map Merging Performance

In this experiment, the same map and trajectories given in Figure 74areused, and the experiment is carried out by changing the velocity sigma value of the second robot from 0.05 to 0.45. Under these conditions, map similarity and the RANSAC success performance is given in Figure 77. The results were obtained from an average of 250 Monte Carlo runs.

Figure 77 shows clearly that the performance of the map similarity decreases from an 80% to a 70% success ratio, and RANSAC is stable at 100%. The stable performance of RANSAC is an expected result of a successful map similarity performance. Therefore, these results are consistent with the simulated dataset results and previous real world dataset results.

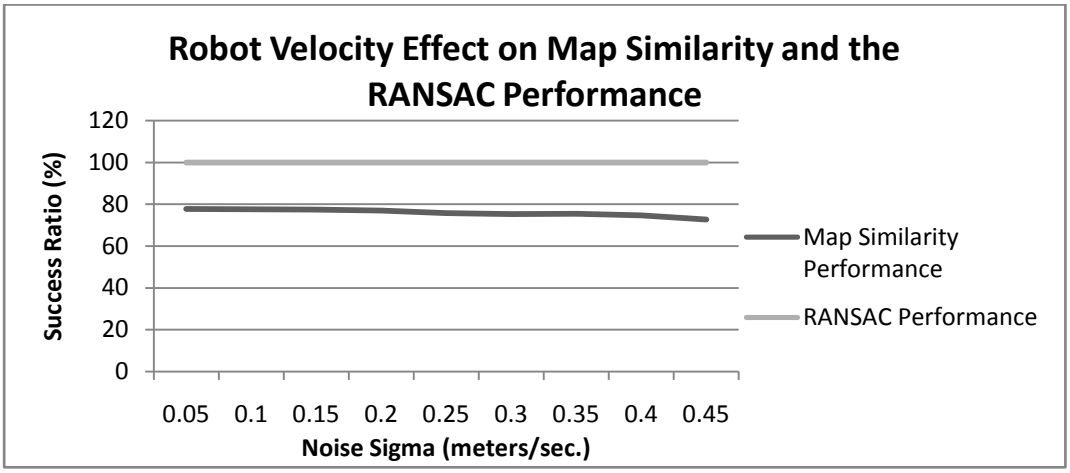


Figure 77 Robot Velocity Noise Effect on Map Similarity and the RANSAC Performance

The results from these experiments for the Covariance Intersection and Modified M-Estimator were similar to the previous experiments. Therefore, the consistent landmark ratio and determinant graphics are omitted in this section. The Normalized Estimation Error Squared results are given in Figure 78. This experiment clearly shows that the NEES of Covariance Intersection algorithm is below the NEES of the Modified M-Estimator.

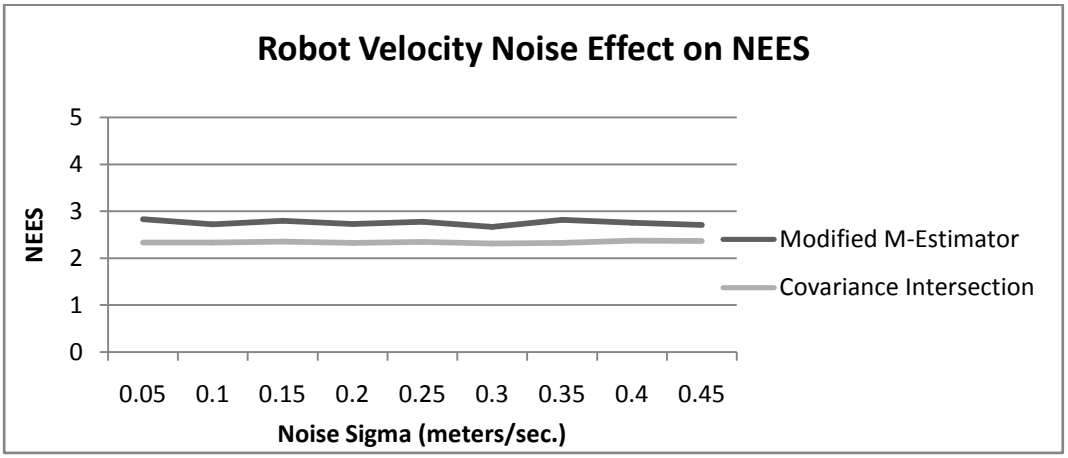


Figure 78 Robot Velocity Noise Effect on NEES

5.3.3. Sensor Range Effect on the Map Merging Performance

In this experiment, the map and trajectories given in Figure 79 are used, and experiment is implemented by changing second sensor range from 9 to 17.

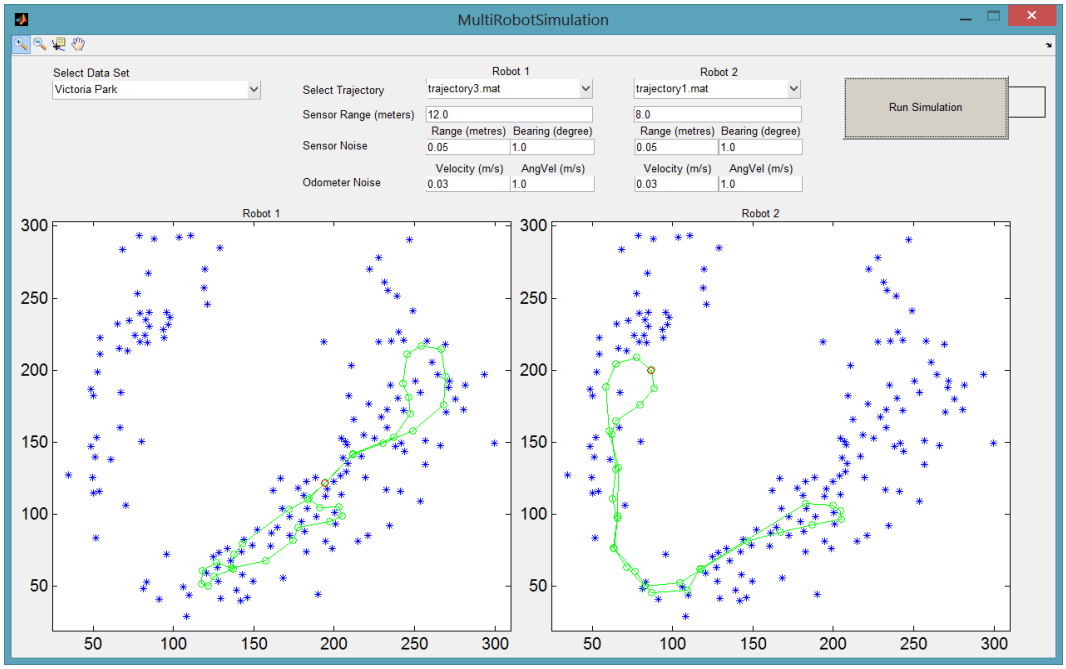


Figure 79 The Victoria Park Extracted Map and Trajectory Illustration

For these conditions, the map similarity and the RANSAC success performance are given in Figure 80; the results were obtained from an average of 500 Monte Carlo runs.

In Figure 80, the performance of map similarity can be clearly seen to be increasing from a 60% to an 80% success ratio, and RANSAC is stable at 100%. The stable performance of RANSAC is an expected result of successful map similarity performance. Therefore, these results are consistent with the simulated dataset results and previous real world dataset results.

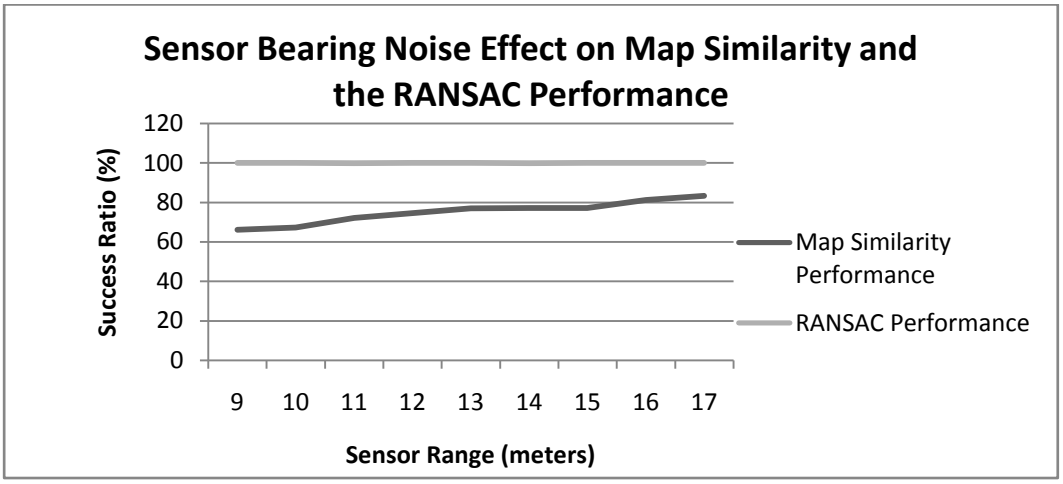


Figure 80 Sensor Range Effect on Map Similarity and the RANSAC Performance

The results from this experiment for the Covariance Intersection and Modified M-Estimator show similar results to the previous experiments. Therefore, the consistent landmark ratio and determinant graphics are omitted in this section. The Normalized Estimation Error Squared results are given in Figure 81. This experiment reveals that the NEES of the Covariance Intersection algorithm is below the NEES of the Modified M-Estimator.

To conclude, these real world dataset experiments show that the RANSAC and map similarity technique used in this thesis give satisfactory results. This is an expected result of RANSAC, because of its outlier elimination property. In fact, RANSAC gives 80% success rate even with the 60% success rate of the map similarity output, as shown in Section 5.2.3. This also reveals that triangle similarity is an appropriate elimination technique to increase the performance of RANSAC even in harsher environments. Moreover, the Covariance Intersection Estimator gives more consistent map merging results than its competitors under

different maps and trajectories with different sensor and robot noise levels. In fact, due its geometric property, which satisfies any degree of correlation between landmarks this is the expected result for the Covariance Intersection.

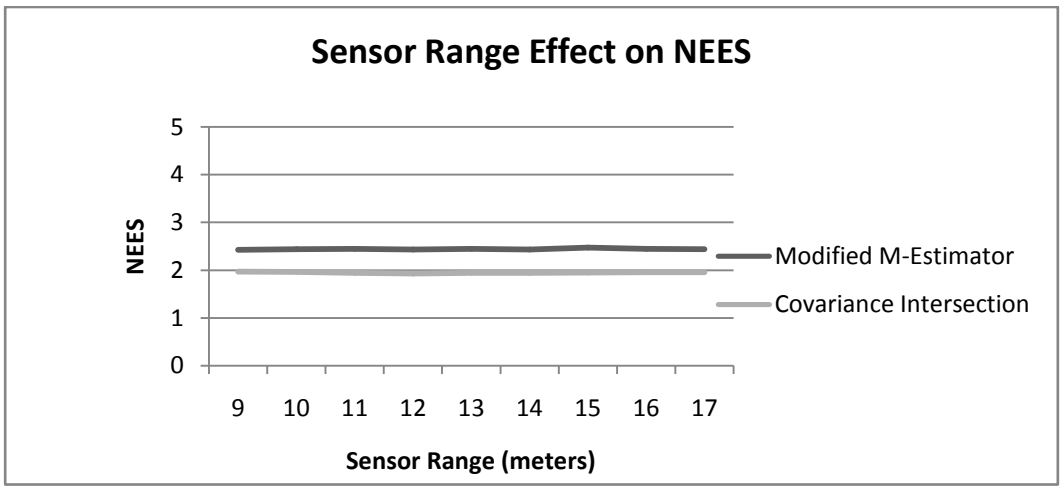


Figure 81 Sensor Range Effect on NEES

5.4. Robustness Analysis of Applied Algorithms

In this section, performance of purposed algorithms, which are Delaunay Triangulation, RANSAC and Covariance Intersection, are examined in terms of their robustness by using real world maps and trajectories. All the algorithms are used with their optimum parameter sets as calculated in section 5.1.

In this experiment, all of the combinations of the trajectories given in Figure 40 are used, and experiment is implemented with the given sensor and robot configuration in Figure 82.

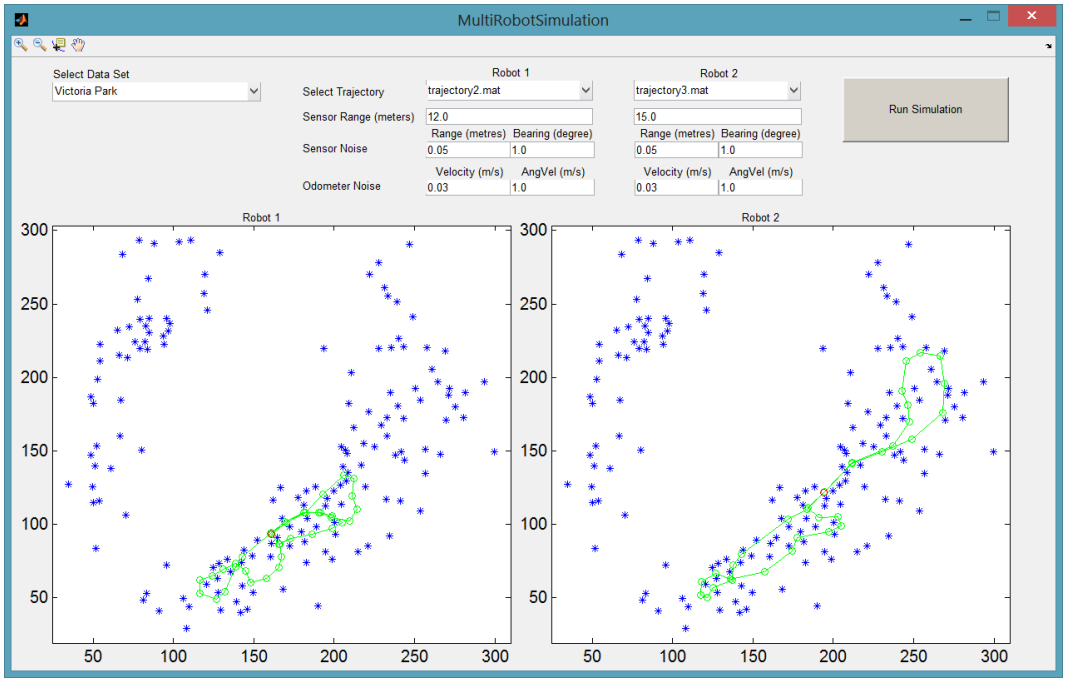


Figure 82 The Victoria Park Map and Trajectory Illustration

For these conditions, the Delaunay Triangulation, map similarity, RANSAC and Covariance Intersection robustness performance is given in Table 7, the results were obtained from an average of 50 Monte Carlo runs, and the ground truth positions of the landmarks are used for the calculation of the NEES values.

Table 7 The NEES values of Purposed Algorithms

Victoria Park Merged Maps	NEES of Robot 1's Map	NEES of Robot 2's Map	NEES of Merged Map
trajectory2 & trajectory1	2,31	4,38	1,54
trajectory2 & trajectory3	2,46	2,64	2,40
trajectory2 & trajectory4	2,92	3,08	2,14
trajectory3 & trajectory1	2,65	3,70	1,84
trajectory3 & trajectory4	2,95	3,13	2,29
trajectory4 & trajectory1	3,14	3,93	2,39

In Table 7, NEES values of landmarks in overlapping region of Robot 1's map and Robot 2's map, and NEES values of merged landmarks are given. This experiment reveals that the NEES values of merged map are always below the NEES of the individual maps of the robots under given experimental specifications.

To conclude, these real world dataset experiments show that, the purposed algorithms used in this thesis give satisfactory results. In fact, merging different sources of information gives more precise and correct results, which is an expected result of map merging process.

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this chapter, the overall summary of the completed work, the contributions of the study, and the experimental results will be presented.

In this thesis, a multi robot SLAM problem is addressed focusing on the map merging issue. The aim is to offer a practical solution to the mapping and localization problem by using the individual robot maps in a team format. This problem is given in detail in Chapter 1. A literature search is undertaken to collate different solutions for the single and multi robot SLAM problems, together with different mapping, data association, multi robot map merging, and map performance evaluation techniques.

Firstly, the single robot SLAM problem is examined and a detailed explanation of the Kalman Filter and its expansions, such as Extended Kalman Filter and Compressed Extended Kalman Filter were given. The Compressed Extended Kalman Filter was selected because it has a low performance cost and satisfactory precision. Moreover, the data association issue was examined with different techniques and their detailed formulations and comparisons are given in the Literature Survey Chapter 2 and the Theoretical Background Chapter 3.

The multi robot SLAM problem is explained by comparison with a single robot case. For instance, extra issues, such as task sharing, communication, and map merging are explained. Moreover, benefits of multi robot solution are also given for comparison. The map merging issue, which is the main purpose of this study, is investigated under different architectures. Centralized, decentralized, and distributed architectures were explained and their advantages and disadvantages were given. Furthermore, a detailed literature survey concerning map-merging issue was presented.

The map merging issue is investigated under two main cases, that of the known relative initial positions of the robots, and the unknown relative positions of the robots. For both of these cases solutions from the literature are presented, and a comparison is given. The unknown initial position, which is a harder problem than known initial position case was selected as a specification of this study. This problem is referred to as map alignment in the literature but in this thesis is called global map transformation. Different algorithms for the solution of this problem are examined in detail; and their detailed formulations are given in theoretical background chapter.

The global map transformation issue is examined through two main approaches. In first approach, the robots estimate the relative position of the other robots using the relative measurement of the robot. For this approach, a detailed formulation for estimation of the transformation matrix is given in the Theoretical Background Chapter 3. For the second approach, the robots share their map information with

their neighbor robots within communication range. For this approach, there are two sub parts; finding the map similarity between transferred landmarks and the search for the transformation matrix. These two issues are explained in detailed and solutions in the literature are given the theoretical background chapter. For solution to the map similarity problem, geometric feature extraction algorithms and similarity calculation techniques, such as the Delaunay Triangulation, topological feature extraction are drawn from the literature and a comparison of their performance is given. For the transformation search, the Adaptive Random Walk, Iterative Translation Search and Random Sample Consensus algorithms are explained and their formulations given.

For final part of the map-merging process, different techniques are used to merge associated (matched) landmark pairs into a more precise single landmark. The result of the literature survey on these techniques is given together with an investigation into the sensor data-fusion issue. To resolve this problem, different techniques such as Maximum Likelihood Estimator, Modified M-Estimator, Covariance Intersection Estimator, Orthogonal Gnanadesikan-Kettenring Estimator, Hybrid Covariance Intersection and Orthogonal Gnanadesikan-Kettenring Estimators are investigated in detail.

For simulation purposes, the open source single robot SLAM with Compressed Extended Kalman Filter is extended to the multi robot SLAM simulator. This extended simulator can load previously generated maps and trajectories, and visualize them. In the SLAM simulation, all this data is hidden from the robot and

they are given as input to the robots with Gaussian distributed noise. The robot's true and estimated positions with covariance ellipse, the true and estimated positions of the measured landmarks with covariance ellipses are visualized by the simulator. All the noise adjustments are enabled by user interfaces, which are implemented in MATLAB GUI from coding created by the author of this thesis. This gives an easy user interaction with the simulator, and the algorithms can be tested using these interfaces. For instance, sensor range, sensor bearing and range noise, robot velocity noise, robot angular velocity noise levels can be adjusted easily by this interface for both robots. For detailed experiments, Monte Carlo analysis option is added to simulation code. Moreover, different real world datasets, such as Central Park, Victoria Park maps and related trajectories are generated manually for realistic experiments. In addition, manual map and trajectory entry ability is added to simulator for new map and trajectory entries.

First part of global map transformation, which is map similarity problem is solved by implementation of Delaunay Triangulation and similarity metric of these triangles. Second part, which is transformation search, is solved with RANSAC implementation. In this implementation, different distance methods and transformation matrix equations are implemented and integrated.

Map merging problem is solved with Maximum Likelihood Estimator, Modified M-Estimator, and Covariance Intersection Estimator implementations.

For evaluation of map-merging algorithms, detailed literature survey is conducted and related works about this study is given in literature chapter. Most suitable

evaluation metrics are implemented for algorithm evaluations in experiments and results chapter. All evaluations of algorithms are done with respect to these metrics.

In experimental part of the study, firstly all algorithms' sensitivity analysis and their run time performances are examined with Monte Carlo runs. The most successful parameter sets of the algorithms are selected by these analyses. Secondly, their performances are compared with simulated map and trajectory data under different conditions. For instance, sensor bearing noise effect, robot's velocity noise effect, sensor range effect, and different trajectory effects are examined by keeping other variable parameters as constant and changing the objective parameter in logical values. Then, similar tests are conducted with real world datasets. All experimental results are represented after running Monte Carlo runs and taking average values of these runs. These experiments show that RANSAC and map similarity technique used in this thesis gives satisfactory results even in harsher environments, which is simulated with real world datasets. Moreover, Covariance Intersection Estimator gives more consistent map merging results than its competitors under different maps and trajectories with different sensor and robot noise levels. Finally, purposed combination of algorithms, which consists of Delaunay Triangulation, RANSAC and Covariance Intersection, is examined with Victoria Park Map and all possible combinations of trajectories. This experiment reveals that robustness of the purposed combination of algorithms.

To conclude, in this thesis the map-merging problem of the multi robot SLAM is examined in detail. Different approaches in literature are presented and compared. The most suitable algorithms are implemented and their formulations are given as theoretical background. For comparisons of implemented algorithms, literature survey is conducted and appropriate evaluation technique is implemented. For simulation environment, multi robot simulator with extended and user-friendly graphical user interface is implemented. In experimental part of the study, algorithms are compared with respect to given evaluation criteria, and their comparison results are presented after the Monte Carlo runs in simulation environment. Moreover, it is conducted that, the combination of algorithms, Delaunay Triangulation, triangle similarity, RANSAC and Covariance Intersection Estimator gives satisfactory results with tolerable time requirements for the solution of distributed map merging problem.

Finally, suggestions concerning future work to extend this study are given below.

- In the current work, global map transformation issue is handled by using a map overlap technique. The detailed formulation of the alternative technique, of using the relative measurements of robots, is given but it is not implemented. In future, the implementation and performance evaluation of these two techniques may be compared.
- In this work, the Delaunay Triangulation and a similarity metric based on these triangles are implemented. However, the literature survey in this

thesis mentions various techniques such as using topological structure of the map. These techniques may be applied and compared.

- The Random Sample Consensus algorithm is used for the multi-robot data association issue in this study. However, the evaluation of alternative techniques such as the Adaptive Random Walk and the Iterative Translation Search may be a beneficial contribution to the SLAM problem.
- Although detailed explanations and comparisons of different map merging architectures are given, only the distributed approach is examined. Therefore, centralized and decentralized architectures may be implemented for the performance analysis and the results analyzed.
- In this study, the Compressed Extended Kalman Filter is used as probabilistic SLAM algorithm. The effect of alternative SLAM algorithms on map-merging performance can also be investigated.

REFERENCES

[1] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45. 2002.

[2] D. Hong-de, D. Shao-wu, C. Yuan-cai, and W. Guang-bin, "Performance Comparison of EKF / UKF / CKF for the," *TELKOMNIKA*, vol. 10, no. 7, pp. 1692–1699, 2012.

[3] J. J. Leonard and H. F. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot," *Proc. IROS '91IEEE/RSJ Int. Work. Intell. Robot. Syst. '91*, 1991.

[4] J. A. Castellanos, J. M. Martinez, J. Neira, and J. D. Tardos, "Simultaneous map building and localization for mobile robots: a multisensor fusion approach," *Proceedings. 1998 IEEE Int. Conf. Robot. Autom. (Cat. No.98CH36146)*, vol. 2, 1998.

[5] J. A. Castellanos, J. D. Tardos, and G. Schmidt, "Building a global map of the environment of a mobile robot: the importance of correlations," *Proc. Int. Conf. Robot. Autom.*, vol. 2, 1997.

[6] J. E. Guivant and E. M. Nebot, "Optimization of the simultaneous localization and map-building algorithm for real-time implementation," *IEEE Trans. Robot. Autom.*, vol. 17, pp. 242–257, 2001.

[7] J. E. Guivant, F. R. Masson, and E. M. Nebot, "Simultaneous localization and map building using natural features and absolute information," in *Robotics and Autonomous Systems*, 2002, vol. 40, pp. 79–90.

[8] E. Ferranti, N. Trigoni, and M. Levene, "Brick&Mortar: An on-line multi-agent exploration algorithm," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2007, pp. 761–767.

[9] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robot. Autom. Mag.*, vol. 13, 2006.

[10] W. H. Huang, "Topological Map Merging," *The International Journal of Robotics Research*, vol. 24. pp. 601–613, 2005.

[11] A. Birk and S. Carpin, “Merging occupancy grid maps from multiple robots,” *Proc. IEEE*, vol. 94, pp. 1384–1397, 2006.

[12] H. Kretzschmar, C. Stachniss, and G. Grisetti, “Efficient information-theoretic graph pruning for graph-based SLAM with laser range finders,” in *IEEE International Conference on Intelligent Robots and Systems*, 2011, pp. 865–871.

[13] B. Kim, M. Kaess, L. Fletcher, J. Leonard, A. Bachrach, N. Roy, and S. Teller, “Multiple relative pose graphs for robust cooperative mapping,” in *Proceedings - IEEE International Conference on Robotics and Automation*, 2010, pp. 3185–3192.

[14] W. Maddern, M. Milford, and G. Wyeth, “Continuous appearance-based trajectory SLAM,” in *Proceedings - IEEE International Conference on Robotics and Automation*, 2011, pp. 3595–3600.

[15] Y. H. Choi and S. Y. Oh, “Grid-based visual SLAM in complex environments,” *J. Intell. Robot. Syst. Theory Appl.*, vol. 50, pp. 241–255, 2007.

[16] R. Sim and J. J. Little, “Autonomous vision-based exploration and mapping using hybrid maps and Rao-Blackwellised particle filters,” in *IEEE International Conference on Intelligent Robots and Systems*, 2006, pp. 2082–2089.

[17] K. L. Ho and P. Newman, “Loop closure detection in SLAM by combining visual and spatial appearance,” *Rob. Auton. Syst.*, vol. 54, pp. 740–749, 2006.

[18] K. M. Wurm, C. Stachniss, and G. Grisetti, “Bridging the gap between feature- and grid-based SLAM,” *Rob. Auton. Syst.*, vol. 58, pp. 140–148, 2010.

[19] I. Tena Ruiz, S. de Raucourt, Y. Petillot, and D. M. Lane, “Concurrent mapping and localization using sidescan sonar,” *IEEE J. Ocean. Eng.*, vol. 29, pp. 442–456, 2004.

[20] J. W. Fenwick, P. M. Newman, and J. J. Leonard, “Cooperative concurrent mapping and localization,” *Proc. 2002 IEEE Int. Conf. Robot. Autom. (Cat. No.02CH37292)*, vol. 2, 2002.

[21] H. F. Durrant-Whyte, “An Autonomous Guided Vehicle for Cargo Handling Applications,” *The International Journal of Robotics Research*, vol. 15. pp. 407–440, 1996.

[22] G. Tuna, K. Gulez, V. Cagri Gungor, and T. Veli Mumcu, “Evaluations of different Simultaneous Localization and Mapping (SLAM) algorithms,” in *IECON Proceedings (Industrial Electronics Conference)*, 2012, pp. 2693–2698.

[23] Uhlmann Jeffrey K., “Dynamic map building and localization for autonomous vehicles,” University of Oxford, 1995.

[24] S. J. Julier and J. K. Uhlmann, “Using covariance intersection for SLAM,” *Rob. Auton. Syst.*, vol. 55, pp. 3–20, 2007.

[25] S. J. Julier and J. K. Uhlmann, “Simultaneous localisation and map building using split covariance intersection,” *Proc. 2001 IEEE/RSJ Int. Conf. Intell. Robot. Syst. Expand. Soc. Role Robot. Next Millenn. (Cat. No.01CH37180)*, vol. 3, 2001.

[26] I. T. Ruiz, Y. Petillot, D. M. Lane, and C. Salson, “Feature extraction and data association for AUV concurrent mapping and localisation,” *Proc. 2001 ICRA. IEEE Int. Conf. Robot. Autom. (Cat. No.01CH37164)*, vol. 3, 2001.

[27] M. Tomono, “Robust 3D SLAM with a stereo camera based on an edge-point ICP algorithm,” *2009 IEEE Int. Conf. Robot. Autom.*, 2009.

[28] H. Andreasson, A. Treptow, and T. Duckett, “Localization for Mobile Robots using Panoramic Vision, Local Features and Particle Filter,” *Proc. 2005 IEEE Int. Conf. Robot. Autom.*, 2005.

[29] Y. Li and E. B. Olson, “Extracting general-purpose features from LIDAR data,” in *Proceedings - IEEE International Conference on Robotics and Automation*, 2010, pp. 1388–1393.

[30] J. Neira and J. D. Tardós, “Data association in stochastic mapping using the joint compatibility test,” *IEEE Trans. Robot. Autom.*, vol. 17, pp. 890–897, 2001.

[31] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, vol. 7. 1973, p. 482.

[32] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*, vol. 9. 2001, p. 584.

[33] A. Marjovi, J. G. Nunes, L. Marques, and A. De Almeida, “Multi-robot exploration and fire searching,” in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, 2009, pp. 1929–1934.

[34] H. Durrant-Whyte, M. Stevens, and E. Nettleton, “Data fusion in decentralised sensing networks,” *... Conf. Inf. Fusion*, pp. 2–7, 2001.

[35] A. Cunningham, M. Paluri, and F. Dellaert, “DDF-SAM: Fully distributed SLAM using constrained factor graphs,” in *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, 2010, pp. 3025–3030.

[36] S. Grime and H. F. Durrant-Whyte, “Data fusion in decentralized sensor networks,” *Control Engineering Practice*, vol. 2. pp. 849–863, 1994.

[37] F. Castanedo, “A review of data fusion techniques.,” *ScientificWorldJournal.*, vol. 2013, p. 704504, 2013.

[38] S. B. Williams, G. Dissanayake, and H. Durrant-Whyte, “Towards multi-vehicle simultaneous localisation and mapping,” *Proc. 2002 IEEE Int. Conf. Robot. Autom. (Cat. No.02CH37292)*, vol. 3, 2002.

[39] E. Nettleton, S. Thrun, H. Durrant-Whyte, and S. Sukkarieh, “Decentralised SLAM with low-bandwidth communication for teams of vehicles,” *Springer Tracts Adv. Robot.*, vol. 24, pp. 179–188, 2006.

[40] L. Carlone, M. K. Ng, J. Du, B. Bona, and M. Indri, “Rao-blackwellized particle filters multi robot SLAM with unknown initial correspondences and limited communication,” in *Proceedings - IEEE International Conference on Robotics and Automation*, 2010, pp. 243–249.

[41] A. Cunningham, K. M. Wurm, W. Burgard, and F. Dellaert, “Fully distributed scalable smoothing and mapping with robust multi-robot data association,” in *Proceedings - IEEE International Conference on Robotics and Automation*, 2012, pp. 1093–1100.

[42] X. S. Zhou and S. I. Roumeliotis, “Multi-robot SLAM with unknown initial correspondence: The robot rendezvous case,” in *IEEE International Conference on Intelligent Robots and Systems*, 2006, pp. 1785–1792.

[43] G. Dedeoglu and G. S. Sukhatme, “Landmark-based Matching Algorithm for Cooperative Mapping by Autonomous Robots,” *Distrib. Auton. Robot. Syst. 4*, pp. 251–260, 2000.

[44] K. Konolige, D. Fox, B. Limketkai, J. Ko, and B. Stewart, “Map merging for distributed robot navigation,” *Proc. 2003 IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS 2003) (Cat. No.03CH37453)*, vol. 1, 2003.

[45] S. Saeedi, L. Paull, M. Trentini, and H. Li, “Neural network-based multiple robot simultaneous localization and mapping,” in *IEEE International Conference on Intelligent Robots and Systems*, 2011, pp. 880–885.

[46] S. Carpin and G. Pillonetto, “Robot motion planning using adaptive random walks,” *2003 IEEE Int. Conf. Robot. Autom. (Cat. No.03CH37422)*, vol. 3, 2003.

[47] S. J. Julier and J. K. Uhlmann, “A non-divergent estimation algorithm in the presence of unknown correlations,” *Proc. 1997 Am. Control Conf. (Cat. No.97CH36041)*, vol. 4, 1997.

[48] P. J. Huber, “Robust Statistics,” *Statistics (Ber).*, vol. 60, pp. 1–11, 2004.

[49] P. J. Huber, *Robust Statistics*, vol. 82. 1981, p. 308.

[50] Sagepub, “ROBUST REGRESSION FOR THE LINEAR MODEL.” [Online]. Available: http://www.sagepub.com/upm-data/17839_Chapter_4.pdf, last visited on October 2014.

[51] A. Zisserman, “Estimators - ML, LS, MAP.” [Online]. Available: http://www.robots.ox.ac.uk/~az/lectures/est/lect34.pdf, last visited on October 2014.

[52] R. Gnanadesikan and J. Kettenring, “Robust estimates, residuals, and outlier detection with multiresponse data,” *Biometrics*, vol. 28, pp. 81–124, 1972.

[53] J. Sequeira, A. Tsourdos, and S. B. Lazarus, “Robust covariance estimation for data fusion from multiple sensors,” *IEEE Trans. Instrum. Meas.*, vol. 60, pp. 3833–3844, 2011.

[54] A. Bredenfeld, U. Visser, I. Noda, Y. Takahashi, P. Fidelman, and W. Nowak, *RoboCup 2006: Robot Soccer World Cup X*, vol. 4020. 2006, pp. 716–723.

[55] A. Finn, A. Jacoff, M. Del Rose, B. Kania, J. Overholt, U. Silva, and J. Bornstein, “Evaluating autonomous ground-robots,” *J. F. Robot.*, vol. 29, pp. 689–706, 2012.

[56] D. Lee, “The map-building and exploration strategies of a simple sonar-equipped robot,” *Disting. Diss. Comput. Sci.*, 19996.

[57] S. Schwertfeger, “Robotic Mapping in the Real World: Performance Evaluation and System Integration.,” *Jacobs Univ.*, 2012.

[58] T. Yairi, “Covisibility-based map learning method for mobile robots,” *Pasific Rim Int. Conf. Artif. Intell.*, 2004.

[59] A. I. Wagan, A. Godil, and X. Li, “Map quality assessment,” *Proc. 8th Work. Perform. Metrics Intell. Syst. - Permis ’08*, p. 278, 2008.

[60] J. Klippenstein and H. Zhang, “Performance evaluation of visual SLAM using several feature extractors,” *2009 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pp. 1574–1581, 2009.

[61] Centre for Autonomous Systems, “The Compressed Extended Kalman Filter (CEKF).” [Online]. Available: http://www.cas.kth.se/SLAM/Presentations/cekf.pdf , last visited on October 2014.

[62] B. Delaunay, “Sur la sphère vide. A la mémoire de Georges Voronoï,” *Bull. l’Académie des Sci. l'URSS*, no. 6, pp. 793–800, 1934.

[63] L. Guibas and J. Stolfi, “Primitives for the manipulation of general subdivisions and the computation of Voronoi,” *ACM Transactions on Graphics*, vol. 4. pp. 74–123, 1985.

[64] S. Fortune, “A sweepline algorithm for Voronoi diagrams,” *Algorithmica*, vol. 2, pp. 153–174, 1987.

[65] Wikipedia, “Delaunay triangulation.” [Online]. Available: http://en.wikipedia.org/wiki/Delaunay_triangulation, last visited on October 2014.

[66] Wikipedia, “Bowyer-Watson Algorithm.” [Online]. Available: http://en.wikipedia.org/wiki/Bowyer–Watson_algorithm, last visited on October 2014.

[67] C. G. A. L. Community, “Computational Geometry Algorithms Library.” [Online]. Available: http://www.cgal.org, last visited on October 2014.

[68] P. Su and R. L. Scot Drysdale, “A comparison of sequential Delaunay triangulation algorithms,” *Computational Geometry*, vol. 7. pp. 361–385, 1997.

[69] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24. pp. 381–395, 1981.

[70] M. Zuliani, “RANSAC for Dummies,” *Citeseer*, 2008.

[71] X. Yu, T. D. Bui, and A. Krzyzak, “Robust Estimation for Range Image Segmentation and Reconstruction,” *{\sc ieee} Trans. Pattern Anal. Mach. Intell.*, vol. 16, pp. 530–538, 1994.

[72] H. Wang and D. Suter, “MDPE: A very robust estimator for model fitting and range image segmentation,” *Int. J. Comput. Vis.*, vol. 59, pp. 139–166, 2004.

[73] R. Iser and F. M. Wahl, “Building local metrical and global topological maps using efficient scan matching approaches,” in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2008, pp. 1023–1030.

[74] P. H. S. Torr and A. Zisserman, “MLESAC: A New Robust Estimator with Application to Estimating Image Geometry,” *Comput. Vis. Image Underst.*, vol. 78, pp. 138–156, 2000.

[75] B. J. Tordoff and D. W. Murray, “Guided-MLESAC: Faster image transform estimation by using matching priors,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, pp. 1523–1535, 2005.

[76] O. Chum and J. Matas, “Matching with PROSAC - Progressive Sample Consensus,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005, vol. 1, pp. 220–226.

[77] J. rí Matas and O. rej Chum, “Randomized RANSAC,” in *Proceedings of the CVWW’02*, 2002, pp. 49–58.

[78] H. Wang and D. Suter, “Robust adaptive-scale parametric model estimation for computer vision,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, pp. 1459–1474, 2004.

[79] M. Zuliani, C. S. Kenney, and B. S. Manjunath, “The multiransac algorithm and its application to detect planar homographies,” in *Proceedings - International Conference on Image Processing, ICIP*, 2005, vol. 3, pp. 153–156.

[80] Z. Zhang, “Parameter estimation techniques: a tutorial with application to conic fitting,” *Image and Vision Computing*, vol. 15. pp. 59–76, 1997.

[81] S. Choi, T. Kim, and W. Yu, “Performance Evaluation of RANSAC Family,” *Procedings Br. Mach. Vis. Conf. 2009*, pp. 81.1–81.12, 2009.

[82] Wikipedia, “Maximum Likelihood.” [Online]. Available: http://en.wikipedia.org/wiki/Maximum_likelihood, last visited on October 2014.

[83] J. Elder, “Multivariate Normal Distribution.” [Online]. Available: http://www.eecs.yorku.ca/course_archive/2012-13/F/4404-5327/lectures/03 Multivariate Normal Distribution.pdf, last visited on October 2014.

[84] K. Senne, “Stochastic processes and filtering theory,” *IEEE Trans. Automat. Contr.*, vol. 17, 1972.

[85] OpenSLAM, “OpenSLAM.” [Online]. Available: http://www.openslam.org/, last visited on October 2014.

[86] Google, “Central Park Map.” [Online]. Available: https://www.google.com/maps/place/Central+Park/@40.7842406,-73.9651565,16z/data=!4m2!3m1!1s0x89c2589a018531e3:0xb9df1f7387a94119, last visited on October 2014.