A Robust and Efficient Autonomous Exploration Methodology of Unknown Environments for Multi-Robot Systems

by

Lillian Goodwin

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

Master of Applied Science in Mechanical Engineering

Faculty of Engineering and Applied Science
University of Ontario Institute of Technology (Ontario Tech University)
Oshawa, Ontario, Canada
May, 2022

THESIS EXAMINATION INFORMATION

Submitted by: **Lillian Goodwin**

Master of Applied Science in **Mechanical Engineering**

Thesis Title: A Robust and Efficient Autonomous Exploration Methodology of Unknown Environments for Multi-Robot Systems

An oral defence of this thesis took place on May 5th, 2022 in front of the following examining committee:

Examining Committee:

Chair of Examining Committee	Dr. Martin Agelin-Chaab
Research Supervisor	Dr. Scott Nokleby
Examining Committee Member	Dr. Meaghan Charest-Finn
Thesis Examiner	Dr. Xianke Lin

The above committee determined that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate during an oral examination. A signed copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

Abstract

Multi-robot systems can provide effective solutions for exploring and inspecting environments where it is unpractical or unsafe for humans, however, adequate coordination of the multi-robot system is a challenging initiative. A robust and efficient methodology for exploration of unknown environments is presented using a k-means method to improve traditional task allocation schemes. The k-means method proposed is an efficient technique due to the algorithm's quick convergence time and its ability to segment a previously unknown map in a logical manner. In this method, a global executive receives frontiers from local robots, filters them, clusters them using the k-means method, and then reassigns them to the agents. A framework is developed in Robot Operating System (ROS) to test the effectiveness of the k-means method. The method is tested over a series of simulations and real-world tests, where it provided significant reductions in exploration time and distance travelled compared to other methods.

Keywords: Multi-Robot Systems (MRS), frontier exploration, K-means, Robot Operating System, Optimization

Author's Declaration

I hereby declare that this thesis consists of original work of which I have authored. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ontario Tech University to lend this thesis to other institutions or individuals for the purpose of scholarly research. I further authorize Ontario Tech University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my thesis will be made electronically available to the public.

Statement of Contributions

Part of the work described in Chapters 3 and 5 has been published as:

L. Goodwin and S. Nokleby, 2021, "Multi-Robot Exploration of Unknown Environments," in *Proceedings of the 2021 CCToMM Symposium on Mechanisms, Machines, and Mechatronics*

At the time of writing, part of the work described in Chapters 3 and 5 has been accepted for publication as:

L. Goodwin and S. Nokleby, 2022, "A K-Means Clustering Approach to Segmentation of Maps for Task Allocation in Multi-Robot Systems Exploration of Unknown Environments," in *Proceedings of the 2022 USCToMM Symposium on Mechanical Systems and Robotics,*

Acknowledgements

I would like to express my deepest thanks to my supervisor Dr. Scott Nokleby for his support, encouragement and time. Your insight and great leadership provided me with everything I needed to flourish throughout my research.

I would also like to thank my family for their belief in me and in my abilities. I would especially like to thank my fiancé, Matthew for your warm encouragement and enthusiasm. Your continual support throughout my life has made all the difference for me in my everyday and in my research.

I would also like to express my gratitude to the DND IDEaS program for the financial support of my research.

I would also like to express my appreciation for the financial support I received from the Ontario Graduate Scholarship (OGS) program.

Table of Contents

Thesis Examination Information ii

Abstract iii

Author's Declaration iv

Statement of Contributions v

Acknowledgements vi

Table of Contents xi

List of Tables xii

List of Figures xv

1 Introduction 1

1.1 IDEaS Project Outline and Goals 2

1.2 Thesis Problem Statement and Goals 3

1.3 Scope 4

1.4 Summary of Contributions 4

1.5 Thesis Outline 5

2 Literature Review 6

2.1 Communication Architectures 6

2.1.1 Centralized Architecture 7

2.1.2 Distributed Architecture 8
2.1.3 Hybrid Architecture 9
2.2 Exploration Methods 10
2.2.1 Frontier Exploration 10
2.2.2 Utility-Based Methods 11
2.2.3 Role-Based Methods 12
2.2.4 Swarm-Based Methods 12
2.3 Utility Model 13
2.3.1 Cost 14
2.3.2 Utility 15
2.3.3 Task Assignments 15
2.4 Map-Merging 18
2.4.1 Online Map-Merging Using a Central Server 19
2.4.2 Online Map-Merging Using Distributed Methods 19
2.4.3 Online Map-Merging Using Hybrid Methods 20
2.4.4 Unknown Initial Positions 21
2.5 Summary 22

3 Methodology **24**
3.1 Framework Overview 24
3.2 Open Source Hardware 26
3.3 Open Source Packages 27
3.3.1 Navigation Stack 27
3.3.2 SLAM 29
3.3.3 Map Merging 29
3.4 Exploration Method 30
3.5 Exploration Assignment 31
3.5.1 Overview of the Assigner Node 31
3.5.2 Filtering Method 33
3.5.3 K-means Method 35

3.5.4 Assignment 37

4 Experimental Design **38**

4.1 Simulation 38

4.1.1 Simulation Environments 38

4.1.2 Simulation Tests 39

4.1.2.1 Bench-Marking Tests 39

4.1.2.2 Scalability Tests 42

4.2 Real-World 45

4.2.1 Real-World Environments 45

4.2.2 Real-World Tests 47

4.2.2.1 Hallway Tests 47

4.2.2.2 Lab Tests 47

4.3 Summary 48

5 Results and Discussion **49**

5.1 Simulation Results 49

5.1.1 Benchmarking Tests Results 49

5.1.1.1 Benchmarking Experiment 1 Results 50

5.1.1.2 Benchmarking Experiment 2 Results 51

5.1.1.3 Benchmarking Experiment 3 Results 53

5.1.1.4 Benchmarking Experiment 4 Results 54

5.1.1.5 Benchmarking Experiment 5 Results 56

5.1.1.6 Benchmarking Experiment 6 Results 58

5.1.2 Benchmarking Tests Discussion 59

5.1.3 Scalability Tests Results 65

5.1.3.1 Scalability Experiment 1 Results 65

5.1.3.2 Scalability Experiment 2 Results 68

5.1.4 Scalability Tests Discussion 70

5.1.4.1 Scalability Experiment 1 Discussion 70

5.1.4.2 Scalability Experiment 2 Discussion 73

5.2 Real-World Results 76
5.2.1 Hallway Tests Results 76
5.2.2 Lab Tests Results 78
5.2.2.1 Lab Experiment 1 Results 78
5.2.2.2 Lab Experiment 2 Results 80
5.2.2.3 Lab Experiment 3 Results 81
5.2.3 Real-World Tests Discussion 82
5.2.3.1 Hallway Tests Discussion 82
5.2.3.2 Lab Tests Discussion 85
5.3 Discussion 86

6 Conclusions and Recommendations for Future Work 89
6.1 Conclusions 89
6.2 Future Work and Recommendations 92

A Experimental Results 101
A.1 Simulation: Benchmarking Tests 101
A.1.1 Benchmarking Experiment 1 102
A.1.2 Benchmarking Experiment 2 103
A.1.3 Benchmarking Experiment 3 104
A.1.4 Benchmarking Experiment 4 105
A.1.5 Benchmarking Experiment 5 106
A.1.6 Benchmarking Experiment 6 107
A.2 Simulation: Scalability Tests 108
A.2.1 Scalability Experiment 1.1 108
A.2.2 Scalability Experiment 1.2 109
A.2.3 Scalability Experiment 1.3 109
A.2.4 Scalability Experiment 1.4 110
A.2.5 Scalability Experiment 2.1 111
A.2.6 Scalability Experiment 2.2 111
A.2.7 Scalability Experiment 2.3 112

A.2.8 Scalability Experiment 2.4 112

List of Tables

4.1 Benchmarking Experimental Design Parameters 40
4.2 Scalability Experiment 1 Design Parameters 42
4.3 Scalability Experiment 2 Design Parameters 44
4.4 Hallway Tests Design Parameters 47
4.5 Lab Tests Design Parameters 48

5.1 Benchmarking Experiment 1 Results 50
5.2 Benchmarking Experiment 2 Results 52
5.3 Benchmarking Experiment 3 Results 53
5.4 Benchmarking Experiment 4 Results 55
5.5 Benchmarking Experiment 5 Results 56
5.6 Benchmarking Experiment 6 Results 58
5.7 Scalability Experiment 1 Results: Far Starting Points 66
5.8 Scalability Experiment 1 Results: Close Starting Points 67
5.9 Scalability Experiment 2 Results 68
5.10 Hallway Tests Results 76
5.11 Lab Experiment 1 Results 78
5.12 Lab Experiment 2 Results 80
5.13 Lab Experiment 3 Results 81

List of Figures

1.1 IDEaS Project Overview 2

2.1 Centralized Architecture 7
2.2 Distributed Architecture 8
2.3 Hybrid Architecture 9
2.4 Inefficiencies in the Nearest Based Allocation Method 11
2.5 Inefficiencies in the Utility-Based Method 14
2.6 Map-Merging Dilemma [37] 19

3.1 Framework Structure Overview [37] 25
3.2 Turtlebot3 Burger Footprint [44] 26
3.3 A Simple Example of Dijkstra's Algorithm [43] 28
3.4 A Simple Example of the DWA Method [43] 29
3.5 Black-box Unmodified Explore_lite 30
3.6 Black-box Modified Explore_lite 31
3.7 Black-box Assigner Node 32
3.8 Assigner Structure 32
3.9 Filter Flowchart [37] 33
3.10 Neighbourhood Sampling 34
3.11 K-means Flowchart [37] 36
3.12 K-means Sample Iteration 37

4.1 Simulation Environments 39
4.2 Benchmarking Tests: Starting Positions 41

4.3 Scalability Experiment 1: Starting Positions 43
4.4 Scalability Experiment 2: Starting Positions 44
4.5 Real-World Environment: Hallway [37] 45
4.6 Real-World Environment: Lab 46
4.7 Real-World Tests: Starting Positions 46

5.1 Benchmarking Experiment 1: Path Comparisons [37] 51
5.2 Benchmarking Experiment 1: Merged Maps [37] 51
5.3 Benchmarking Experiment 2: Path Comparisons 52
5.4 Benchmarking Experiment 2: Merged Maps 52
5.5 Benchmarking Experiment 3: Path Comparisons 54
5.6 Benchmarking Experiment 3: Merged Maps 54
5.7 Benchmarking Experiment 4: Path Comparisons 55
5.8 Benchmarking Experiment 4: Merged Maps 55
5.9 Benchmarking Experiment 5: Path Comparisons 57
5.10 Benchmarking Experiment 5: Merged Maps 57
5.11 Benchmarking Experiment 6: Path Comparisons 58
5.12 Benchmarking Experiment 6: Merged Maps 59
5.13 Benchmarking Experiments: Average Time Comparisons 60
5.14 Benchmarking Experiments: Percent Improvement K-means Time . . 61
5.15 Benchmarking Experiments: Average Distance Comparison 62
5.16 Benchmarking Experiments: Percent Improvement K-means Distance 62
5.17 Benchmarking Experiments: Standard Deviation Time 63
5.18 Benchmarking Experiments: Standard Deviation Distance 63
5.19 Benchmarking Experiment 1: Map Coverage Comparisons [37] 64
5.20 Scalability Experiment 1a: Merged Maps for Far Starting Points . . . 66
5.21 Scalability Experiment 1a: Paths for Far Starting Points 66
5.22 Scalability Experiment 1b: Merged Maps for Close Starting Points . . 67
5.23 Scalability Experiment 1b: Paths for Close Starting Points 67
5.24 Scalability Experiment 2: Merged Maps 69

5.25 Scalability Experiment 2: Paths 69
5.26 Scalability Experiment 1: Average Time Comparison 70
5.27 Scalability Experiment 1: Average Distance Comparison 71
5.28 Scalability Experiment 1: Standard Deviation Time Comparison . . . 72
5.29 Scalability Experiment 1: Standard Deviation Distance Comparison . 73
5.30 Scalability Experiment 2: Average Time 74
5.31 Scalability Experiment 2: Average Distance 74
5.32 Scalability Experiment 2: Standard Deviation Time 75
5.33 Scalability Experiment 2: Standard Deviation Distance 75
5.34 Hallway Experiment 1: Merged Maps [37] 77
5.35 Hallway Experiment 1: Path Comparisons [37] 77
5.36 Lab Experiment 1: Merged Maps 79
5.37 Lab Experiment 1: Path Comparisons 79
5.38 Lab Experiment 2: Merged Maps 80
5.39 Lab Experiment 2: Path Comparisons 80
5.40 Lab Experiment 3: Merged Maps 81
5.41 Lab Experiment 3: Path Comparisons 82
5.42 Hallways Experiments: Time 83
5.43 Hallway Experiments: Average Distance 83
5.44 Hallway Experiments: Map Coverage Comparisons [37] 84
5.45 Lab Experiments: Time Comparisons 85
5.46 Lab Experiments: Average Distance Comparisons 86

6.1 Examples of K-means Segmentation of the Map 91

Chapter 1

Introduction

Unmanned Grounds Vehicles (UGVs) have become a daily part of many peoples lives, using them as vacuums for cleaning their houses [1], as transportation agents for manufacturing environments [2], and even as autonomous disinfecting agents for fighting COVID-19 [3]. UGVs can provide practical solutions for inspecting and exploring dangerous environments where humans are unable to go, or providing quicker, more efficient inspections and exploration where it might take humans a long and tedious time. Even more flexible is the use of coordinated robotic teams to complete tasks. For example, in a Search and Rescue (SAR) scenario a team of robots could be deployed into an unknown environment to quickly search the area for survivors, damage to buildings, or other tasks with little to no control being provided by an operator. The full autonomy of the system can prove advantageous in remote areas where there is limited communication. However, there is little research and implementation on the use of robotic teams in unknown environments.

In this thesis, a framework is developed for use in a Multi-Robot System (MRS) to efficiently explore unknown environments autonomously and with scalable robotic teams. This framework allows the MRS to enter into an unknown environment and

map the area in a robust and efficient manner.

1.1 IDEaS Project Outline and Goals

This thesis is being developed for a project with the Department of National Defense (DND) investigating the development of effective human-machine cooperation with autonomous systems. The motivation behind this project is to further the design of Intelligent Adaptive System (IAS) for Autonomous Unmanned Vehicle Systems (AUVS). The project is broken into five main areas of research: trust modelling, trust experimentation, trust management, intelligent automation, and interface adaptation. In Figure 1.1 an overview of the different tasks associated with this project are visualized.

As a part of the IDEaS project intelligent automation is one of the five areas of

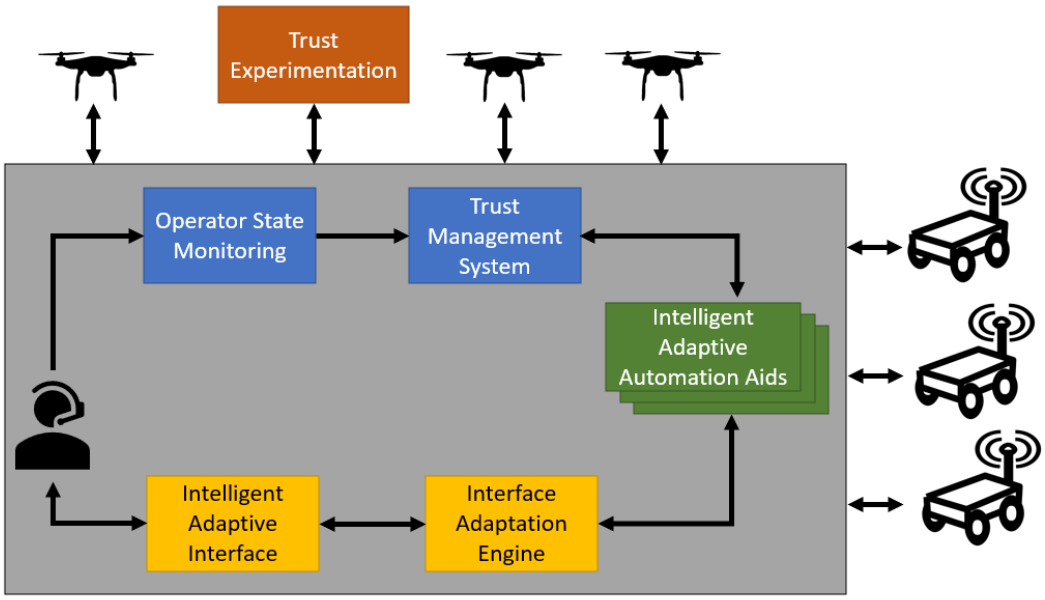


Figure 1.1: IDEaS Project Overview

focus. This area of focus includes the improvement of the current state-of-the-art for MRS. For this thesis the focus is on the intelligent automation side of this project

specifically in UGV robotic teams, this includes the use of a MRS for exploration of unknown environments.

1.2 Thesis Problem Statement and Goals

The problem addressed in this thesis is the development of a framework for a MRS that can be used to explore an unknown environment in a time-efficient and adaptable manner. This larger goal was separated into different smaller goals listed below. A detailed description of the tasks is explained in the paragraphs following.

- Determine a novel, robust and efficient method to facilitate a cooperative exploration task. This task includes a method for segmentation of an unknown map and task allocation.
- Develop the new method using a programming platform to facilitate the proposed new method.
- Develop a framework to facilitate the testing of the newly developed method.
- Test the method in simulated environments.
- Test the method in real-world environments.

For the first goal, MRS exploration of unknown environments is a well researched topic in the literature. However, it is difficult to develop a system that facilitates a well coordinated exploration that improves the time for the exploration task, the overlap in map coverage and efficient task allocation methods. A new MRS exploration method is to do be developed to take into account the primary factors of a time-sensitive exploration, scalable and coordinated MRS exploration task.
Next, the proposed MRS exploration method will be implemented into a software environment. This is challenging as it requires the use of effective real-time programming. This is because algorithms can take a long time computationally and this makes implementation into real-time applications ineffective.
The third goal is to then develop a framework for testing in the Robot Operating

System (ROS) environment. This requires the manipulation of many different open source packages and heavy modification of some other packages.
The last two goals relate to testing and bench-marking the new MRS exploration method in both simulated and real-world environments.

1.3 Scope

The scope of this thesis is the design and implementation of a MRS framework for the exploration of unknown environments, to develop the system to minimize the distance travelled by each individual robot, to minimize the overlap in map coverage, and to minimize the execution time. The framework should be scalable and capable of handling different environments and numbers of robots.

1.4 Summary of Contributions

The main contributions of this thesis are:

- Development of a time-sensitive, scalable and coordinated MRS exploration using a novel application of the k-means method.
- Development of a new method in C++ to facilitate k-means segmentation of a map in a MRS.
- Development of a ROS framework to facilitate the implementation of the new method.
- Simulated experimental testing of the new MRS exploration method.
- Real-world experimental testing of the new MRS exploration method.
- Benchmarking of the current state-of-the-art with the new MRS exploration method.

1.5 Thesis Outline

The remainder of this thesis is as follows: Chapter 2 presents a review of the state-of-the-art in regards to exploration tasks in MRS. This includes the review of different task allocation methods. Chapter 3 includes an in-depth presentation of the methodology utilized for this system, including open source packages used and new methods developed. Chapter 4 consists of a review of the different experiments and why they were conducted, as well as hypothesized results. Chapter 5 includes the experimental results for all the different experiments conducted and a detailed discussion of the results. Chapter 6 concludes the findings of this thesis and suggests future work for the development of this research.

Chapter 2

Literature Review

This chapter consists of a review of the state-of-the-art with regards to MRS autonomous exploration of unknown environments. For the purposes of this literature review, the review has been split into five different sections: a review of different architectures for communication in MRS, a review of different methods for exploration, a review of frontier exploration and the utility model, and a review of map merging techniques in MRS.

2.1 Communication Architectures

The method of communication used between the agents in a MRS is called the communication architecture. When working in MRS there are three different architectures that can be used to facilitate communication including centralized, distributed, and hybrid architectures.

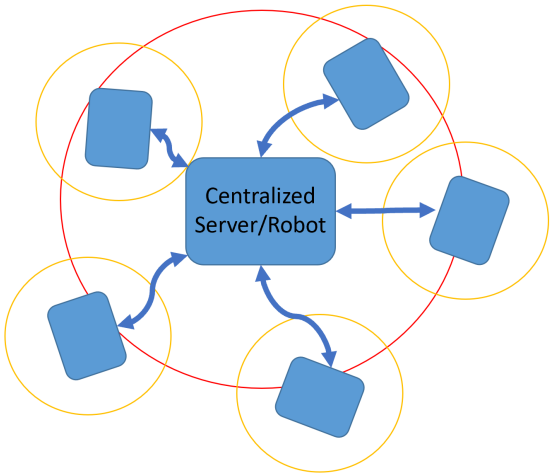


Figure 2.1: Centralized Architecture

2.1.1 Centralized Architecture

As seen in 2.1, a centralized architecture relies on one computer/robot to control all of the other robots' movements, data processing, etc. In the figure shown the blue boxes represent individual agents, the red and orange circles represent the communication range of the central server and agents, respectively, and the blue arrows represent the flow of data of the central server and agents. While this can be an efficient algorithm for a few robots, the addition of more robots into a system can become problematic with increased communication overhead and possible information losses [4, 5]. A centralized approach also results in a lack of redundancy and a single point of failure in the MRS. Recent work has proposed the usage of a centralized server system to communicate with embedded system robot nodes [4]. This approach allows for lower level tasks to be computed in an embedded system, such as local path planning, drivers for sensors, etc., while the upper level tasks, such as complex computation and visualization, are handled on the server. This is advantageous but still suffers from the single point of failure issue.

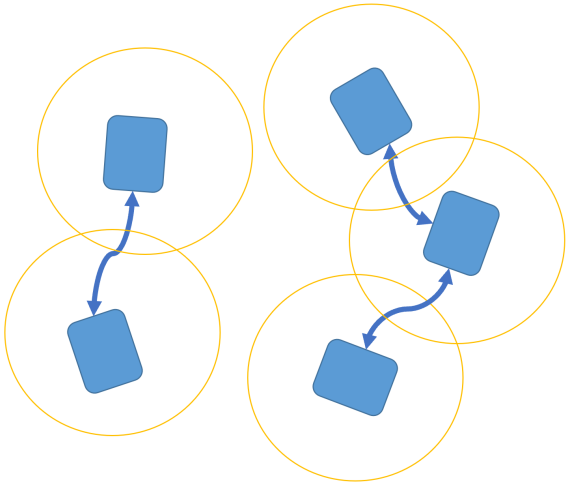


Figure 2.2: Distributed Architecture

2.1.2 Distributed Architecture

Distributed architectures rely solely on the individual robots to do all of the path planning, information processing, etc., without any guidance from a master robot. As can be seen in Figure 2.2, the blue boxes represent individual agents, the orange circles represents the communication range of the agents, and the blue arrows represent the flow of data between the agents in communication range. The communication happens between all the individual robots; this requires the need for complex and robust communication protocols/bandwidth [4]. This is advantageous because the MRS is non-reliant on a single point of contact which adds to the robustness of the system as a whole. However, this can become challenging as coordination of the MRS becomes much more complex. Some examples of distributed approaches are Yamauchi's frontier exploration [6] and Particle Swarm Optimization (PSO) used for exploration [7].

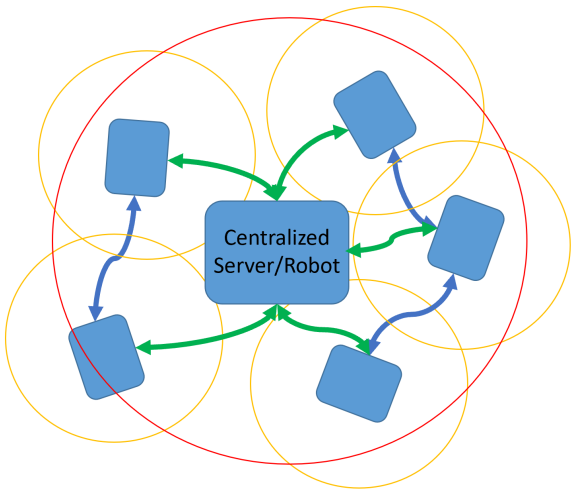


Figure 2.3: Hybrid Architecture

2.1.3 Hybrid Architecture

The final architecture is a hybrid architecture which allows for the communication between a master robot and between all the individual robots. This can be pictured in Figure 2.3, where the blue boxes represent individual agents, the red and orange circles represent the communication range of the central server and agents, respectively, and the green and blue arrows represent the flow of data of the central server to agents, and agent to agent, respectively. This can become advantageous for applications such as role-based exploration, like that of Hoog and Cameron, where communication between individual robots and to a central server is needed [8]. This can become advantageous in providing a more robust system without a single point of failure, but limits the system in relying on a central server/robot.

As shown, there are three main different architectures used for communication in MRS. Centralized architecture is advantageous as it allows for easier coordination of the MRS and real-time global map construction, but this method introduces a single point of failure. Distributed architectures do not rely on a single point of contact which solves the single point of failure issues, but require more complex communication protocols. Hybrid architectures can have applications in communication

restricted environments, offer easier global map construction, but the architecture does not have the ability to create a real-time global map. The choice in architecture highly depends on the type of exploration and Simultaneous Localization and Mapping (SLAM) algorithm chosen for the exploration application, which will be explained in the next sections.

2.2 Exploration Methods

For the purposes of this thesis, MRS exploration is defined as the task of placing multiple robots in an unknown area and having them adequately search and map the area autonomously. Different methods exist in the literature to facilitate this exploration task. In this section, a brief review of frontier exploration and some other exploration method variations are reviewed.

2.2.1 Frontier Exploration

Frontier-based methods run on the concept of exploring an environment based upon the boundary of explored and unexplored space [9]. This allows robots to navigate to a centroid of the border of an unexplored region using shared maps [10]. A simple strategy for frontier-based area coverage is the nearest based allocation method as proposed by Yamauchi in 1997 [9]. Yamauchi utilizes a completely distributed architecture where each robot will share their local sensor information to produce frontier lists. This method is attractive because it does not require centralized communication to coordinate the robots, making the system more fault tolerant [11]. However, this is not the most efficient method as multiple robots may navigate to the same frontier [5] as can be seen in Figure 2.4.

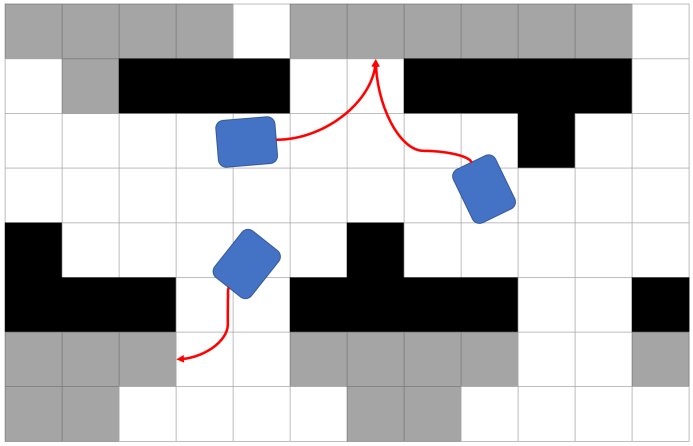


Figure 2.4: Inefficiencies in the Nearest Based Allocation Method

2.2.2 Utility-Based Methods

To solve the frontier assignment issues in the previous approach, utility-based navigation models were developed for frontier assignment using a cost-based analysis. Simmons proposed a method where the robots make bids to a central computer based upon adding the information gain and subtracting the cost [10]. Recognizing the reliance on a centralized method, Zlot et al. minimized the need for a centralized server and utilize a distributed cost-utility or market-based approach [12]. The utility model is advantageous compared to the nearest frontier model in preventing navigation to the same points, but still requires overlap in area coverage at times [5]. This method can also become computationally inefficient as the number of robots and frontiers increases because the algorithm performs iteratively. Seeking to minimize the overlapped area, a rank-based frontier allocation method is proposed by Bautin et al. to not only take into account the distance to the frontier, but also the distribution of the robots among the frontiers [11]. An extension of the rank-based method in [13] also keeps track of the frontiers within a certain threshold distance to ensure all frontiers are explored.

2.2.3 Role-Based Methods

In SAR applications, limited communication can also be a factor as investigation of remote areas may not allow for heavily communicative reliant methods to be deployed [8]. To address this, an area of interest has become the exploration of environments in restricted communication areas. Role-based exploration was mainly developed to mitigate the problem of limited communication within an exploratory robotic network [14]. Rooker and Birk, proposed a method to ensure robots did not travel outside of communication range [14]. They did this by assigning penalties in a utility function for a robot outside of communication range. Another role-based solution that emerged proposed a modification to sequential greedy assignment, where the decision making for assignment of frontiers is distributed to several different robot planners [15]. This method is advantageous for scalability applications, but still the research assumes a completely connected network among the agents. In another role-based work, robots are assigned roles of either explorer or relay [8]; the explorer can go beyond the wireless communication limits and return to meet relays at previously defined rendezvous points. Still another application looks to reduce the communication burden by proposing a role-based method where only the frontier points are shared with a leader [16]. This leader is in charge of assigning goals to other robots called explorers.

2.2.4 Swarm-Based Methods

Another exploration strategy is the leader-follower strategy. Leader-follower methods focus on the team and its role as opposed to the structure of the environment itself [5]. Desai proposed decentralized control to coordinate robots by assigning a group leader and individually controlling all the other robots to follow in formation [17]. Only the group leader in this case knows the trajectory information. Reynolds work with

motion coordination for animals is the starting place for most flocking based MRS exploration [18]. Reynolds developed flocking behavior based upon the principles of matching velocity, flock centring and avoidance of collisions. This approach has been expanded to exploration in a couple of sources where a hybrid frontier-based and flocking-based strategy are used [19], [20]. Inspired by flocking and limited communication with large robotic teams, swarm robotics approaches have also been used in MRS exploration [7], [21, 22]. The method as presented by Wang et al. [7] uses a distributed frontier-based algorithm with PSO. In this approach, the map is broken into sub-map areas where the robots will not enter an area already covered; the exploration state is used to explore its own sub-area and the walking state uses PSO to determine paths based upon already explored areas and the positions of the other robots. More recently Kumar et al. presented a SAR application based upon Darwinian PSO and simulations within the ROS [23].

In summary, the starting point for most exploration techniques has been the frontier-based method as proposed by Yamauchi [9]. Since then different methods for the exploration of unknown areas using MRSs have emerged, namely the utility-based method, role-based method, swarm-based method, and more. The focus of this thesis is on utility-based exploration methods to develop a more efficient exploration task.

2.3 Utility Model

Based upon the inefficiencies in Yamuachi's distributed model, the need to coordinate the robotic exploration becomes apparent as seen in the previous section. Simmons et al. [10] first developed a utility-based analysis for assigning tasks to each of the agents. However, using this method requires fine-tuning of the cost, utility, and goal assignments to provide an efficient exploration process. In Figure 2.5, it can be seen that the two agents in the system travel to two frontiers closest to them both while a

large frontier at the top of the map is being left unexplored as a result of some utility based methods. In this section an in-depth review of the utility model outlining the

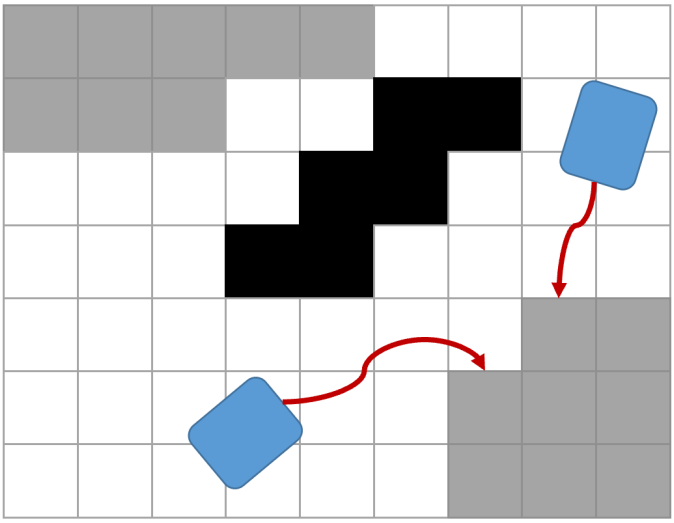


Figure 2.5: Inefficiencies in the Utility-Based Method

definition of cost, utility and task assignment methods will be conducted.

2.3.1 Cost

Determining the cost of reaching a frontier cell is important because it allows for the quantification of resources used in order to reach a frontier. The majority of methods in frontier exploration seem to have a similar approach when dealing with the calculation of costs, namely calculating the resources based upon distance travelled to reach the frontier. There are some slight variations found within the literature such as in [24], where Burgard et al. computed the cost using a dynamic programming algorithm called value iteration. They defined the cost using an iterative approach, that considers the cost as equivalent to the distance to the cell multiplied by the occupancy value. In this method convergence is guaranteed, as long as the cost is not negative and the environment is bounded. Fox et al. [25] utilized a similar approach for cost, seeking to find the minimal cost path using an A* search algorithm. Another

less utilized approach is mentioned by Zlot et al. [12] calculated the costs based upon the time estimate to reach a frontier, as opposed to the distance. In [26], the cost to reach a point is modified to not only calculate cost based upon the distance, but also to take into account the energy consumption associated with travelling to that point.

2.3.2 Utility

The utility of frontier cells is a quantifiable representation of the usefulness associated with a robot reaching a specific frontier cell. The utility is said to be an estimate of the information gain expected when reaching a certain cell. Burgard et al. [24] computed utility by starting off each frontier cell with the same utility value and changing the utility based upon the target selected points for each robot. Their utility value also takes into account the probability of a sensor covering the range of the adjacent cells as to avoid overlap in coverage. In work by Rooker and Birk [27], a proposed solution in communication limited environments included a constraint to the utility function to not allow a loss of communication between all robots to the base station. Later Rooker and Birk expanded their research to robotic packs [28], as to remove the need for connectivity to a base station. Fox et al. [25] defined utility by taking into account the unknown area visible from the frontier as a way to estimate the total utility.

2.3.3 Task Assignments

Selecting goal points for each robot becomes necessary when dealing with robots in teams. The utility model basic algorithm for assigning tasks usually consists of determining the costs associated with reaching a goal point and subtracting that value from the utility associated with reaching that point. Different methods are used for assignment and for optimizing the assignments. Following after Yamauchi [6], in

Simmons work [10] the idea of constructing bids was utilized. Bids were quantified by subtracting the cost from the utility, as stated above. Tasks are then assigned by the central executive. The first robot is awarded the task who had the highest net utility and then the central server subtracts that robot's bid from all other frontiers. This process iterates until all the frontiers are assigned. Zlot et al. [12] build upon Simmons bidding approach and applied a market economy approach to task allocation. This method improves the performance and optimization of the task assignment by changing the negotiations to local robots. Gerkey and Mataric [29] followed Zlot's work and utilized an auctioneer type of system to manage the assignment of the frontiers, this includes anonymous broadcasting in order to communicate with the other robots and eliminates the need for the operations executive. This system works more as a fully distributed method. However, it was found that the dynamic system is very sensitive to changes in the environment. Burgard et al. [24] used a while loop approach to optimize the task assignment until each robot has been assigned a frontier point. The utility associated with each goal point is updated after each assignment of a target point. Burgard et al. also looked into how to apply this algorithm into limited communication environments. Following a bidding approach, Sheng et al. [30] developed a system that seeks to optimize the distributed approach for goal assignment. They used a similar broadcasting type of framework as in [29], but now they make no assumption of connectivity to the same network. The bidding works by broadcasting to robots within its local network a bid for a specific amount of time, if no other robot joins the session and beats the bid, the robot wins that bidding session. If there is new map information gained within that time constant, the bids are re-evaluated after the maps have been updated. This process repeats until no more frontier cells are left. Building on the utility based model for frontier navigation, a rank-based allocation method is proposed by Bautin et al. [11], which seeks to further optimize not only the utility of each robot, but the distribution of the robots along each of the frontiers.

Another cost-based method used for assigning roles is the Hungarian method as defined by Kuhn [31]. This method finds the optimal solution for role assignment given

the costs in an nxn matrix. In step 1 the matrices are modified to compute a reduced cost matrix, subtracting the minimal element in each row and then repeating for the columns. In step 2 the minimal number of lines to cover all of the zeros in the matrix is found, if the number of lines is equal to the dimension, n, of the matrix, then optimal values can be found. If this is not the case, step 3 requires the minimum value in the matrix to be subtracted from each value in the matrix not covered by the lines and added to each value covered by both horizontal and vertical lines. Steps 2 and 3 are repeated until the number of lines equals the dimension, n. In work by Wurm et al. [32] targets for MRS exploration are assigned using the Hungarian method. In their work, not only are frontiers considered, but segmentation of the map is introduced in order to facilitate more time efficient exploration.

K-means has also been used for task assignment in some areas of the literature. In work by Elango et al. [33] k-means is used to balance task allocation by clustering different tasks together based upon distance and then assigning them using an auction based mechanism. Solanas and Garcia [34] used k-means in frontier exploration to cluster frontier cells of an occupancy grid into different clusters to be assigned to agents. The number of clusters are determined to be equal to the number of robots within the system. Each robot is assigned to its closest cluster. In work by Puig et al. [35] an optimization algorithm is applied to the the k-means exploration algorithm to ensure optimal task assignments. K-means is also used in work by Faigl et al. [36] and is used to solve a travelling salesman problem for MRS exploration. From these publications, k-means has been proven to show improvement in task allocation and in MRS exploration. In [34–36], the assumption is made that each agent knows a global map when clustering frontier points and ignores the process of merging information from local agents to a global centralized server. This assumption is not practical when dealing with many agents in real-time applications. These methods do not consider how a global map is being created, how each of the local agents know their position within the global map, how communication from the agents in the system to centralized server effects the exploration process, and how their method can be used in real-time applications.

In summary, within frontier navigation, to facilitate efficient coordination in MRS, there must be a defined cost, utility and goal point assignment method. Costs often are represented as a function of the resources needed to reach a frontier cell or as a function of the distance travelled. Utility is defined as the expected information gain associated with reaching a frontier cell. The goal point assignment is a form of a multi-objective optimization problem being carried out in real-time and the different techniques employed greatly change the efficiency of the system. However, within goal point assignment a main distinction exists regarding whether the assignment of goals is done for the global system or local sub groups of the robotic team.

2.4 Map-Merging

How to facilitate the making of a global map can become one of the most challenging topics in multi-robot exploration in unknown environments. This is because the robots must not only know their position within their own map through SLAM, but also must have some representation of their relative positions in relation to each other or the world to facilitate online map-merging. In addition, when mapping an environment a robot has to deal with two types of uncertainty or noise within the data, the noise from the sensors and the uncertainty in the odometry [24]. In Figure 2.6 robots are pictured within a map, and different laser scans of the environment are taken. The dilemma shows that the when merging two maps into one global map, the maps must be transformed into the correct coordinate frame. If map-merging can be done offline, different techniques to merge the maps exist, often based upon graphical methods, such as in [38]. Chang et al. merge individual graphical maps by adding the defined edges from the different maps. For the purposes of this review it is assumed online map-merging is the desired approach, with an emphasis on real-time map updates.

In the following sections, map-merging techniques will be explored, including, online map-merging using a central server, a decentralized approach, a hybrid approach, and

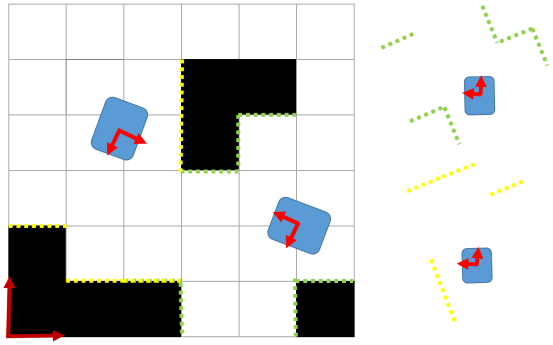


Figure 2.6: Map-Merging Dilemma [37]

some techniques managing unknown initial positions.

2.4.1 Online Map-Merging Using a Central Server

One technique commonly used for map-merging within an online approach is proposed by Simmons et al. where each robot builds its own local map, a centralized server receives the local map and then builds them into one global map using a maximum likelihood estimate [10]. The drawback is that this method rests on the assumption that each robot will begin the exploration task within view of each other and a given relative global location.

2.4.2 Online Map-Merging Using Distributed Methods

A distributed approach to map-merging relies on each individual robot to store their own maps of the environment and will facilitate sharing of information to reduce inefficiency in exploration. This approach starts off most basically like that in [6]. In Yamauchi's approach each robot holds their own global evidence grid, this grid contains information about the environment they currently have, and every time a robot arrives at a frontier the local evidence grid is constructed and broadcast to the other robots. The robots will then take this information and update their global evidence grids. Sheng et al. [30] utilize a scheme that includes each robot storing both

a raw map table and a local map. This raw map table contains the raw information obtained by each robot and the order of the map data obtained, as to not double the data transfer. The local map is constructed using a map fusion mechanism. Sheng et al. assume that the relative position of each robot is known and the robots start at positions close to one another. Fox et al. [25] does not assume known relative locations and implements a system where robots exchange laser range scans and odometry information whenever they are in communication range. They then use a particle filter to estimate their relative locations. In this method, maps are merged using probabilistic constraints. Another method, as proposed by Meier et al. [39], deals with low bandwidth constraints and reduces the amount of communication necessary by approximating the map broadcast as a polygonal shape.

2.4.3 Online Map-Merging Using Hybrid Methods

A hybrid approach for map-merging often facilitates individual map-merging, as well as the end goal of returning all of the information to a centralized server to build one global map. Zlot et al. [12] make use of explicit map sharing between the robots' local maps, and as mentioned before, utilize an operations executive to facilitate the merging of the total global map. To do this, the operations executive sends out a request for the map data to be sent out and all robots within communication range send their local maps back. However, the assumption made is that the relative orientations of the robots are known in regards to one another, this follows from the earlier assumption in their work that the robots must be kept a reasonable distance from one another [12]. Hoog et al. [40], in their role based approach, utilized an explorer/relay system. Their proposal for map-merging consisted of merging maps based upon their relative location through their own localization, but accounts for some error in localization when the explorer and relay meet a rendezvous point.

2.4.4 Unknown Initial Positions

Based upon the sections above, it can be seen that most map-merging techniques seem to favour knowing the initial positions due to the decreased complexity. As mentioned in the previous section, Fox et al. [25] looked into the possibility of unknown initial position. Some other techniques have been explored, such as in [41], where map-merging was made using two steps. The first step happens when two robots come within communication range they then estimate their location in the other robot's map by matching the received map to the patch from the other robot. The second step requires the robot to verify the validity of the hypothesis by attempting to meet the robot at an expected location based upon the merged map, if they do not meet as expected the hypothesis is rejected. In work by Zhou and Roumeliotis [42] a rendezvous solution is utilized. In their solution, when robots come within communication range a relative pose and distance measurement are taken to compute the transformation matrix between the maps.

In summary, map-merging using centralized methods is defined by the use of a central server to merge all of the local maps into one global map. Distributed methods usually consist of broadcasting information to other robots in their communication range to update their own local maps. Hybrid methods leverage the use of local robots broadcasting information to each other, but still pursue the need for a central server to receive the total global map at the end. Finally, the majority of these methods count on the availability of the initial positions of each robot [6, 10, 12, 40], but sometimes this is not the case. If there are unknown initial positions, rendezvous methods [42], probabilistic methods [25], and hypothesis based methods [41] can be used to determine the localization of each robot.

2.5 Summary

In the literature three different types of architectures are presented, namely centralized, distributed, and hybrid methods. Centralized methods present advantages for real-time online approaches and when an operator is being kept in the loop of control such as in military applications. Distributed methods provide a more robust exploration as each agent only requires itself to perform a successful exploration. Hybrid methods allow for a global representation to a centralized server, but also are not susceptible to the single point of failure issues that centralized methods have. However, distributed methods do not provide information to a central agent as needed for this project and hybrid methods require complicated, robust communication protocols which introduces a large chance of error.

Many exploration techniques exist such as the frontier-based method [9] and its variations like the utility model [10], role-based methods, and unique methods such as swarm-based methods and more. For the purposes of this thesis the utility model is explored.

The utility model defines a bid as the expected information gain upon reaching a frontier subtracting the cost associated with reaching this frontier [10]. The model then solves a multivariate optimization problem to assign goal points to the different agents, this could be achieved by iterative assignment methods [10], local negotiation [12,29,30], or with other techniques such as the Hungarian method [32]. K-means has been shown to provide a more efficient task allocation scheme as seen in [34–36]. However, these works assume each agent has access to a known global map and ignore the real-time process of map merging and its effect on k-means task allocation.

Map-merging techniques utilized within the literature were reviewed for their application to frontier-based exploration. It was found that map-merging often assumes initial relative positions or global positions are known [6, 10, 12, 40]. However, some techniques exist that try to eliminate the need for initial relative locations to always be known [25, 41, 42].

In SAR applications, some of the important variables become, a time efficient and robust exploration process, the availability of real-time updates to a central server, and accurate map-merging for in-depth views of disaster environments. Under these assumptions for the purposes of this thesis frontier exploration in a MRS is conducted using a centralized communication architecture, k-means clustering to optimize for task assignments using the utility model and map-merging using known initial positions. K-means is utilized without making assumptions about each agent having access to a global map and considering the effect the communication architecture has on a real-time exploration process.

Chapter 3

Methodology

In this chapter the methods used for the novel MRS exploration framework are presented. First, the proposed framework is presented. Next, a review of the open source hardware and packages utilized within the framework are covered. Then, the novel methods developed for this work are presented. Parts of the methodology are also presented in work by Goodwin et al. [37, 43].

3.1 Framework Overview

An overview of the framework and nodes running on the central computer and an individual agent's on-board computer is shown in Figure 3.1. In this figure, the TurtleBot3 (TB3) has an embedded node which runs on the on-board Raspberry Pi computer. This embedded node is running the TB3 node for bringing up the robot. The bring up node sets up different publishers and subscribers for the robot. It publishes the sensor information, battery state, transformations, odometry information and more. It subscribes to topics such as the velocity commands, motor power, and more. The centralized server runs five different types of nodes: the modified ex-

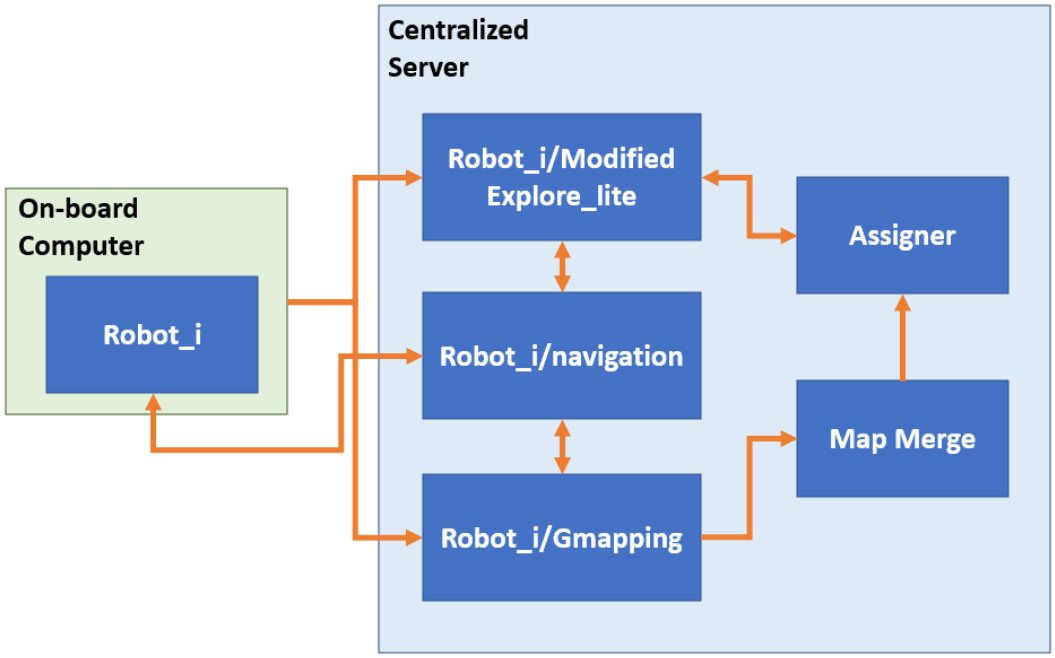


Figure 3.1: Framework Structure Overview [37]

plore_lite nodes, the navigation nodes, the gmapping nodes, the map merge node and the assigner node. The modified explore_lite node runs an instance for each robot in the MRS. The modified explore_lite node takes in the occupancy map information from an agent and finds frontiers. The frontier points are grouped into frontier regions and published to a topic. The explore_lite node also subscribes to a goal points topic. The navigation node runs an instance for each agent. The navigation node receives goal points for an agent and then provides global and local path plans to facilitate the autonomous navigation of an agent to the proposed goal point. The gmapping node runs an instance for each agent. The gmapping node takes in laser scans and odometry information from an agent and then creates an occupancy map. The map merge node runs one instance for the MRS and uses estimated poses to create a global map by combining multiple local agents' maps. The assigner node runs one instance for the MRS and takes in all of the frontier region centroid points published by each agent in the system. The assigner node clusters the points into clusters using a k-means method and then publishes the goals to a topic.

3.2 Open Source Hardware

For the purposes of this thesis open source robots namely TB3 Burgers were used along with a PC. The TB3 Burger is a small mobile robot designed to run on ROS. More information can be found at: https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/. The TB3 Burgers are differential drive robots featuring an on-board Raspberry Pi model 3 B+, CR driver board, dual motors, and a Light Detection and Ranging (LiDAR) sensor. The footprint for the TB3 Burger can be seen in Figure 3.2. As seen in previous work by Goodwin and Nokleby [43], the

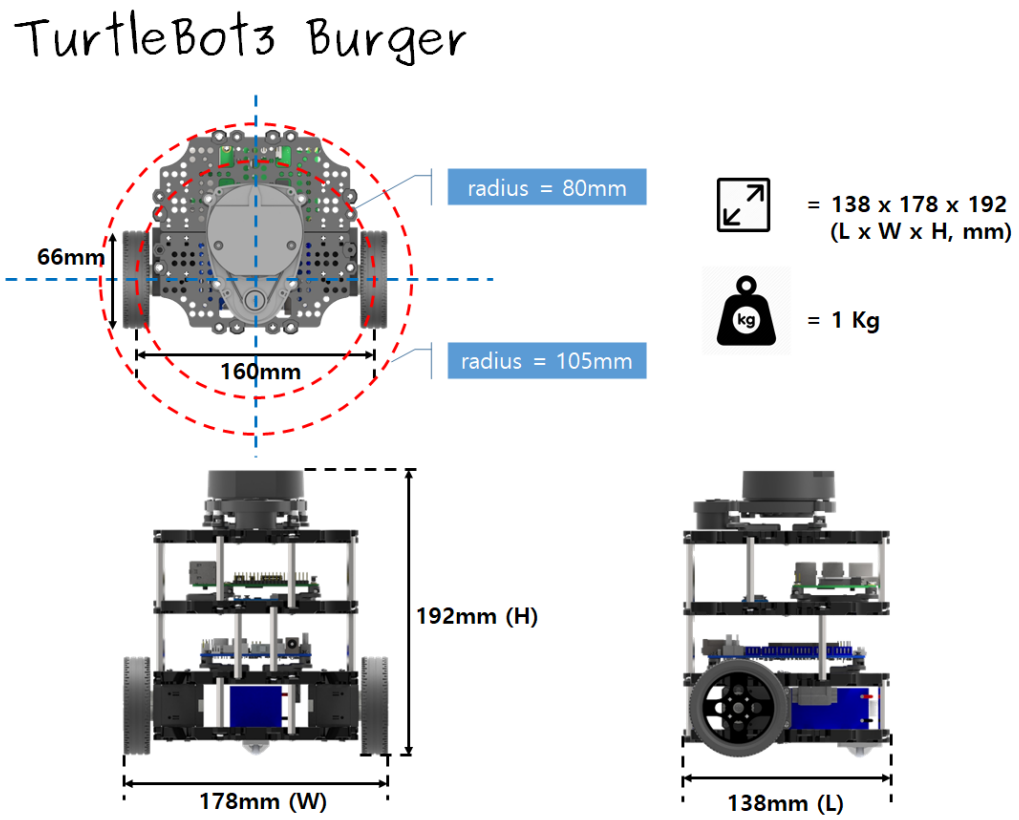


Figure 3.2: Turtlebot3 Burger Footprint [44]

equations of motion can be defined as:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{\cos(\theta)}{2} & \frac{\cos(\theta)}{2} \\ \frac{\sin(\theta)}{2} & \frac{\sin(\theta)}{2} \\ \frac{1}{l} & \frac{-1}{l} \end{bmatrix} \begin{bmatrix} v_R \\ v_L \end{bmatrix} \tag{3.1}$$

where $\dot{x}$, $\dot{y}$, and $\dot{\theta}$ are the translation and angular velocities of the robot, θ is the heading angle, and l is the distance between wheels. For the TB3s, l is 160 mm.

3.3 Open Source Packages

The development of this work was done in ROS version Kinetic Kame. ROS is a series of open source packages created for the development of robotic control and simulation. The TB3 has a set of open source packages pre-configured for use with ROS and were downloaded for use in this thesis [45]. In this section a review of the different open source packages used will be conducted.

3.3.1 Navigation Stack

The navigation stack is a set of packages designed for motion and path planning for mobile robots. In this thesis, the package move_base from the navigation stack [46] is used after goals have been set to determine a global and local path plan for autonomous navigation. The libraries of this package allow for the choice of different kinds of global and local path planning.

The global path planner chosen is entitled Navfn in the navigation stack [46]. Navfn seeks to find the lowest cost to travel from the robot's current location to the goal

point. The code implements a variation of Dijkstra's algorithm [47]. The algorithm finds the minimum cost by using a graphical technique which represents different points on a map as nodes. The algorithm keeps track of the shortest distance from each node to the starting point. Upon every iteration, the algorithm visits the current lowest cost node and visits all the unvisited neighbours to that node updating the current lowest cost node. It repeats this pattern until the shortest path to the goal is found. A basic example of Dijkstra's algorithm can be seen in Figure 3.3.

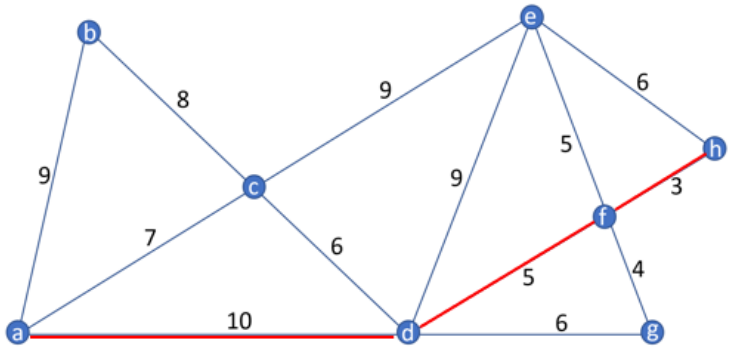


Figure 3.3: A Simple Example of Dijkstra's Algorithm [43]

The local path planner chosen is the dwa_local_planner from the navigation stack [46]. This algorithm uses the Dynamic Window Approach (DWA) as proposed by Fox et al. [48]. The main purpose of a local path planner is to control the robot to follow the global path. In the DWA approach this is achieved by treating the local path plan as an optimization problem, i.e., trajectories are limited to circular trajectories able to be reached within a short period of time. The robot is also assumed to be circular. A series of forward simulations are performed and illegal trajectories are discarded, meaning trajectories that would cause a collision. The best path is chosen by optimizing the multi-objective function that considers the local path that is closest to the global path, stays furthest away from all obstacles present, remains closest to the goal, and achieves the fastest velocity. In Figure 3.4 a simplified graphic for the DWA planner is shown. In the figure, possible trajectories are in blue, illegal trajectories are in red and the global path plan is in green.

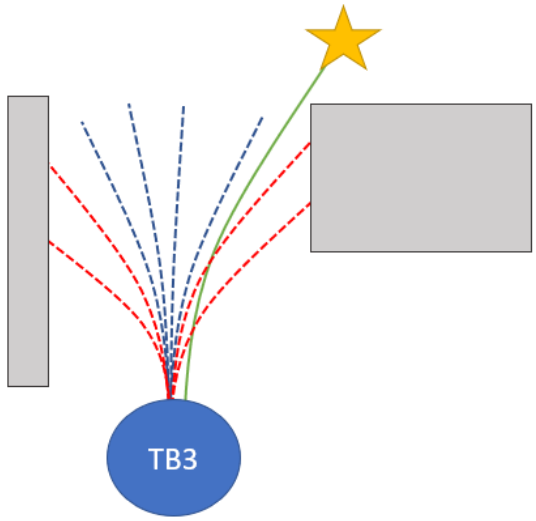


Figure 3.4: A Simple Example of the DWA Method [43]

3.3.2 SLAM

Gmapping is a package in ROS used for Simultaneous Localization and Mapping (SLAM) [49]. The package is written to take in the robot's odometry information and laser range information from the LiDAR. The package then uses a probabilistic approach to determine the occupancy of each grid cell from the readings. A transformation from the base link of the robot to the LiDAR range scanner is also needed to determine these maps.

3.3.3 Map Merging

The multi_robot_map_merge by Hörner [50] is a package that takes in multiple agents' maps and merges them into one global representation. There are two assumptions that can be made to use this package the first is that the global or relative starting points of all the agents in the system are known and the second is the global or relative starting points are unknown.

If the global or relative starting positions are known this package uses a simple stitching algorithm to merge the two maps together. The algorithm in this case takes in

multiple occupancy grids and merges them together using rigid transformations given from the initial positions.

If starting positions are not known this package uses a more intense feature matching algorithm to determine the merged map. This algorithm uses a heuristic approach where a feature detector is used to try and find matches using pairwise matching. If enough features are matched, a transformation will be estimated and applied if the confidence is high enough.

3.4 Exploration Method

In the new MRS exploration framework as described in section 3.1, one package was modified at the source to provide the function needed. A detailed explanation of the package's original purpose and modified purpose will be reviewed.

Explore_lite is also a package created by Hörner [50]. This package's intention is to facilitate a single robot exploration, where frontiers are assigned greedily until there are no more frontiers on the map. In Figure 3.5 a black-box version of the original explore_lite algorithm is shown. In the original explore_lite package an occupancy

Figure 3.5: Black-box Unmodified Explore_lite

grid and occupancy grid updates topic from the SLAM node is needed, as well as the robot's global position. The algorithm then finds frontiers and prioritizes them based upon distance and size. A centroid location of a grouping of frontiers is sent to the move_base node producing the movement commands as the output.

In the modified explore_lite the goals are not being set by the node itself, instead it is receiving goals from a published topic and the list of centroids of frontier regions

are being published to a topic. In Figure 3.6 the black-box version for the modified explore_lite can be seen, the changes are in yellow.

Figure 3.6: Black-box Modified Explore_lite

3.5 Exploration Assignment

The novel contribution of this thesis is the novel method for exploration entitled assigner developed for MRS exploration. This assigner node was first introduced in other work by Goodwin and Nokleby [37]. In this method, a version of k-means is used to determine task allocation for the MRS. The use of k-means in MRS exploration is not novel [33–36], but the approach given in this research is. In this work, k-means is used to group centroids from regions of frontiers, not group frontier cells as in the work by Solanas and Garcia [34]. Additionally, the k-means algorithm is run when updates to local maps are passed to the central server and compared to a global map. A global map is not assumed available to each agent as in other works [34, 36], therefore this method is suitable for real-time and real-world applications. For the purposes of this section, centroids of regions of frontiers will be referred to as Points of Interest (PoI). In this section, details of the assigner node will be presented.

3.5.1 Overview of the Assigner Node

The assigner node has the basic function of taking in all new local PoI information from each agent in the MRS and then grouping these points into n, number of robot clusters. These clusters are then assigned to the closest robot in the system. In Figure

3.7 the basic black-box function of the assigner node can be seen. The assigner method

Figure 3.7: Black-box Assigner Node

takes in three topics: the published PoI from each agent, the locations of each of the robots, and the global map topic from the multi_map_merge node. The basic structure of this node can be seen in figure 3.8. In this figure one can see three main distinctive

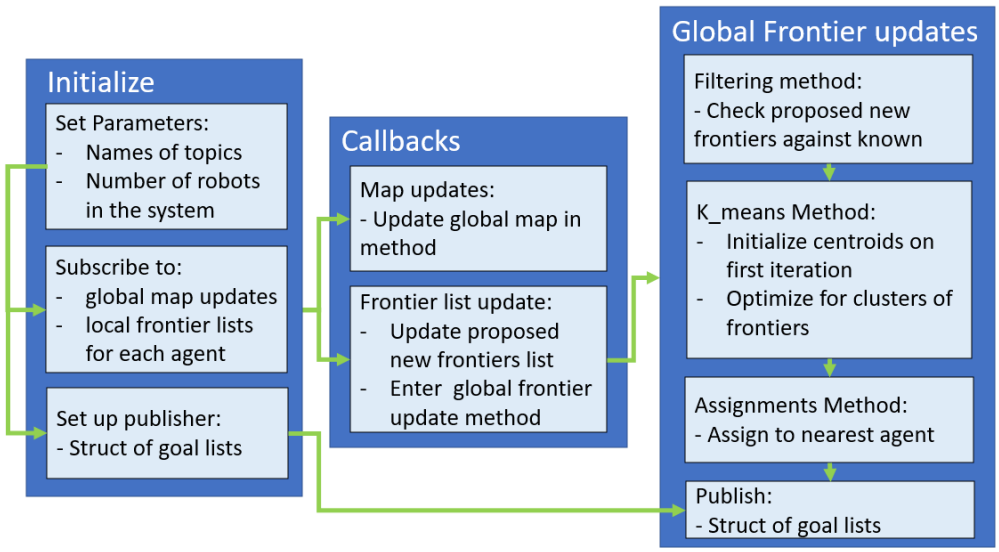


Figure 3.8: Assigner Structure

portions of this node. The first is to initialize the parameters, subscriptions and publishers. The main parameters used in this system are for setting the names of different topics and how many agents will be a part of the system. Subscriptions are needed for the global map topic and for the PoI lists. The subscriptions for the PoI lists are performed in loops where the names are modified depending on the number of robots in the system. The publisher is a customized structure of messages filled with vectors of goal lists. The number of goal lists in the structure is dependent on the number of robots in the system. Each robot is initialized with a unique ID number starting at 1, 2, 3, ...n and this is used to access the correct goal list in

the structure. The second portion of this node is for the different callback methods. Callback methods are used in real-time operations to update the global map topic when the multi_map_merge package provides updates and to update the proposed new frontier vector when local agents find new PoI. The third portion of this node is the method for updating the global frontier list. The basic function of this portion of the node serves to filter proposed new PoI against a global map, cluster the globally unknown PoI, and then assign these clusters to the nearest robot or agent. This portion will be discussed further in the next section.

3.5.2 Filtering Method

The filtering method serves to check PoI coming from each agents local map and determine which points are unknown globally. If the points are unknown globally they are added to a list of global points of interest (i.e., the PoI list). In Figure 4.6(b) the algorithm for filtering the new PoI is detailed. The psuedocode, as seen in previous

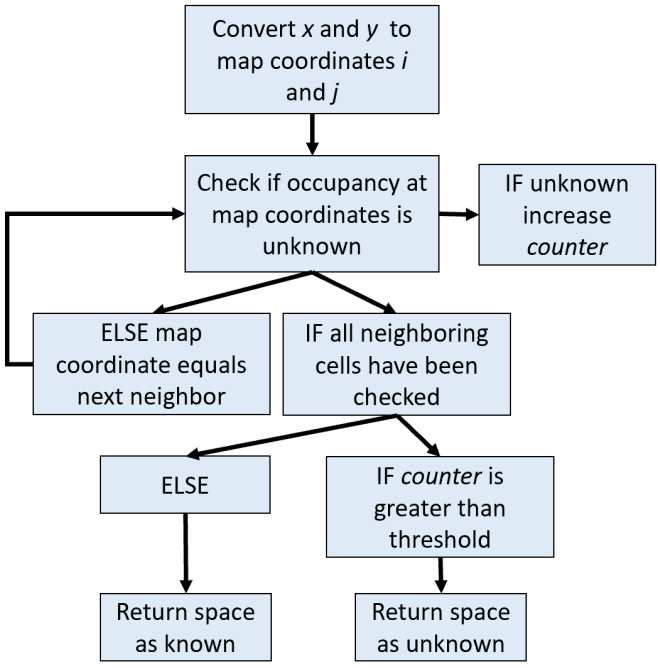


Figure 3.9: Filter Flowchart [37]

Algorithm 1 Filter [37]

```
for i ← −n : n do
    for j ← −n : n do
        x and y global coordinates converted to occupancy coordinates k and r
        if Occupancy_Map(k + i, r + j) ← unknown then
            Counter ← Counter + 1
        end if
    end for
end for
if Counter < Threshold then
    Remove from frontier list
end if
```

work by Goodwin and Nokleby [37], is also presented in this thesis in Algorithm 1. The algorithm is detailed as follows, the point of interest is first converted from world (x, y) coordinates to map coordinates (i, j). Then the occupancy at the map coordinates is checked for a neighbourhood of values. A neighbourhood of values in this case refers to the closest 24 squares region or a 5x5 square region. In Figure 3.10 an example of this sampling is shown. In the figure, the blue dot represents

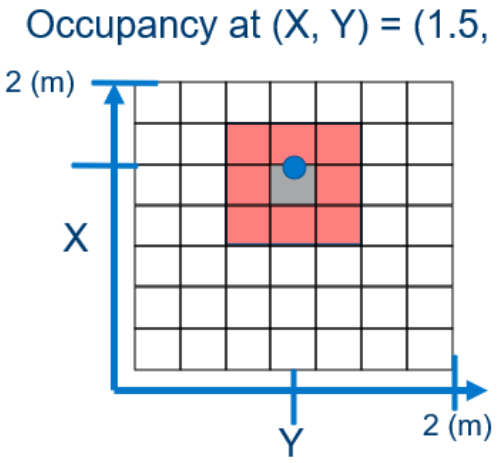


Figure 3.10: Neighbourhood Sampling

the conversion of the (x, y) coordinates to the map coordinates (i, j). The grey grid square represents the closest square grid to the coordinate. The red squares represent the closest 3x3 square values or the closest 8 neighbours to the converted coordinate. Considering only one grid value can often lead to inaccurate conclusions as to the

status of an area. The purpose in sampling an area is to gain a more accurate picture as to whether a point is unknown globally or not. Through experimentation, increasing this to a square neighbourhood more accurately shows whether the point is truly known globally or not. The size of the grid chosen for the occupancy map can also change the size of the neighbourhood needing to be sampled. The smaller the grid, the bigger the neighbourhood would have to be to get an accurate picture and vice versa. Through experimentation for a grid scale of resolution 0.05 m/cell (meters per cell), it was found that if out of 25 (5x5) occupancy values greater than 11 of them are unknown, it is very likely this spot is unknown on the global map. In binary occupancy grids if the position is unknown it will equal a value of -1. So, for every cell in the 5x5 neighbourhood region the occupancy is checked and if it equals a value of -1 a counter will be increased. If at end of checking all of the neighbours the counter value is greater than or equal to 11, then the point is determined to be unknown. If the counter value is less than 11 the point is determined as already known and deleted from the PoI list.

3.5.3 K-means Method

After the vector of unknown PoI is returned, a k-means algorithm is then run. K-means was first introduced by Macqueen [51], a modified and simple version is used in this framework. The basic function of this method is to optimize for n clusters from a given data set. The pseudocode, as presented in other work by Goodwin and Nokleby [37], can be seen in Algorithm 2.

The algorithm works as follows, for each system the initial cluster centroids are chosen at random to be different entries in the vector of globally unknown PoI. The number of clusters is chosen to equal the number of robots given in the system for easy assignment. After the initial cluster centroids are chosen, each point in the vector is assigned to its nearest cluster. The new centroid of each cluster is found by averaging the x and y values. This is evaluated for a certain number of generations/epochs

Algorithm 2 K-Means Clustering of PoIs [37]

```
Random points from the PoI list are chosen as centroids
Centroids ← random(PoI_list)
for i ← 0 : epochs do
    for i ← 1 :size(Centroids) do
        for j ← 1 :size(PoI_list) do
            Find the closest centroid to each PoI on the list
            if dist(centroid(i), PoI_list(j) then
                PoI_list(j).cluster ← i
            end if
        end for
    end for
    Find sum of x and y values for each cluster
    for j ← 1 :size(PoI_list) do
        SumX(clusterID) ← PoI_list(j).x
        SumY(clusterID) ← PoI_list(j).y
    end for
    Find the new x and y values for the centroids
    for j ← 1 :size(PoI_list) do
        Centroid.x ← SumX(clusterID)/number_points
        Centroid.y ← SumY(clusterID)/number_points
    end for
end for
```

or until a termination criteria is met. In Figure 3.11 a simple flowchart can be seen detailing the algorithm. An example of the algorithm is shown in Figure 3.12,

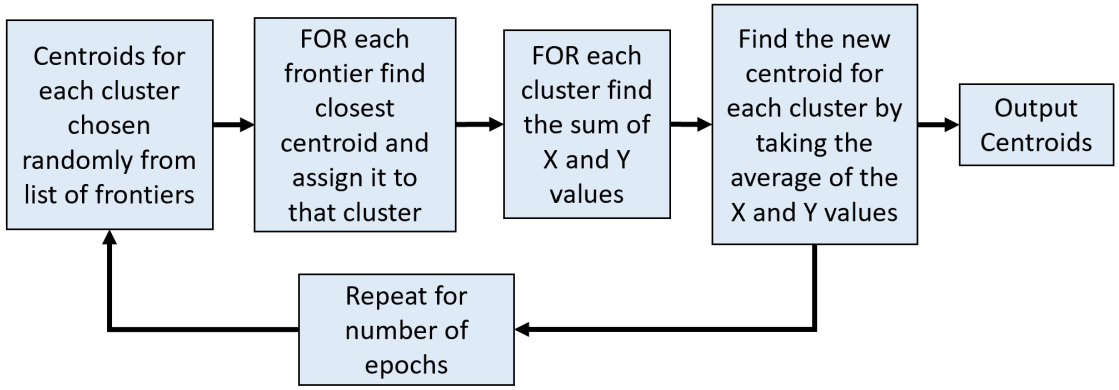


Figure 3.11: K-means Flowchart [37]

where one can see the agents in navy blue with several points of interest in light blue

around them. A random centroid is chosen to be the orange point and each of the blue points are assigned and pointing to their cluster centre. After optimization the cluster centres have moved to the gray point in the second part of the image and then as per the method outlined in this thesis they are assigned to the nearest agent.

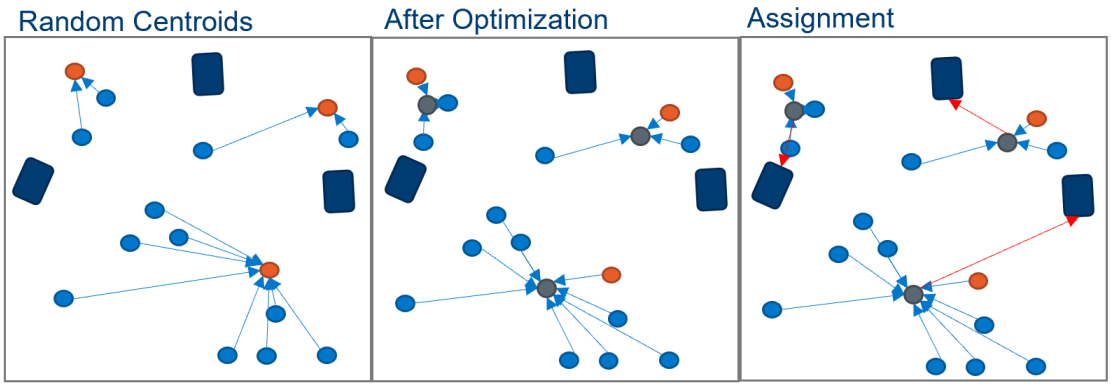


Figure 3.12: K-means Sample Iteration

3.5.4 Assignment

The assignment method used for the sake of this methodology is simply an iterative assignment method. The reason this was chosen is for the simplicity in computational load for the system. The iterative assignment simply assigns the nearest cluster to each agent starting with the lowest agent ID to the highest ID and each agent can only be assigned one cluster at a time.

Chapter 4

Experimental Design

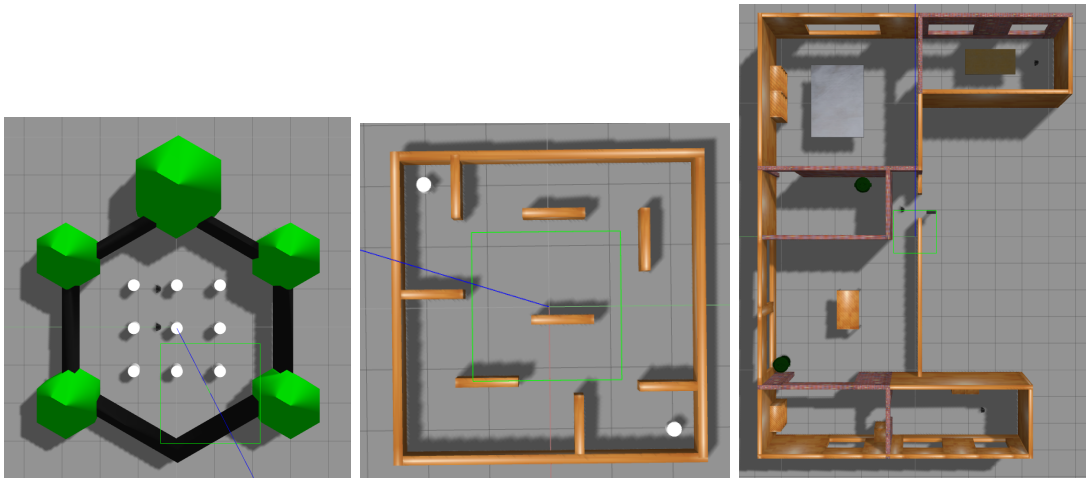
In this chapter a review of the different experiments and the parameters used for the evaluation of the new framework is explored. The experiments are broken into two different sections: simulation and real-world experiments.

4.1 Simulation

For the simulation portion of the experiments Gazebo [52] is used as the environment. A few different open source environment models are used along with different size robotic teams. Average results are recorded in a table in the main body of the thesis. Full results can be found in Appendix A.

4.1.1 Simulation Environments

In Figure 4.1, one can see the three different environments being used for the tests. The first is the TB3 World environment this is used for a small team of robots. The second is the medium size TB3 Stage 4 environment that is more complex than the first environment. The last is the larger TB3 House environment used to determine



(a) TurtleBot3 World (b) TurtleBot3 Stage 4 (c) TurtleBot3 House

Figure 4.1: Simulation Environments

the versatility of the algorithm. The second and third environments are used for testing variable size robotic teams. All of these environments are from the open source TB3 Gazebo environment models [45].

4.1.2 Simulation Tests

The types of tests for this framework can be broken into two different categories: benchmarking tests and scalability tests.

4.1.2.1 Bench-Marking Tests

For the benchmarking tests in this system, there are three different methods evaluated to create a baseline of whether the proposed framework is performing efficiently.
The first method consists of an uncoordinated experiment where multiple robots explore the environment but have no communication with each other. This method serves as a baseline for determining how long it would take an individual robot to perform the same task on average. The second method utilizes a shared map. The

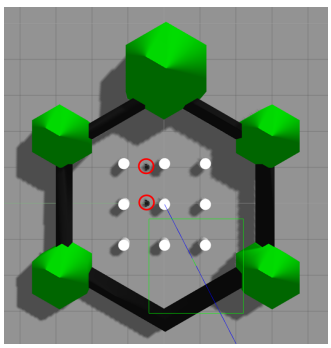
Table 4.1: Benchmarking Experimental Design Parameters

Experiment Number	**Environment**	**Number of Robots**	**Starting Points**
1	TB3 World	2	Fig. 4.2(a)
2	TB3 Stage 4	2	Fig. 4.2(b)
3	TB3 Stage 4	3	Fig. 4.2(c)
4	TB3 Stage 4	3	Fig. 4.2(d)
5	TB3 Stage 4	4	Fig. 4.2(e)
6	TB3 House	3	Fig. 4.2(f)

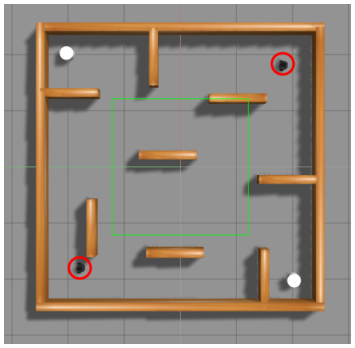
idea behind this approach is to show how MRS exploration would react if only the map is shared and there is no method to coordinate the exploration between the different agents. This method serves to show the benefit of the proposed k-means algorithm for task assignment over having a method with only a shared global map. The last method considered is the new proposed k-means method.

For the benchmarking tests, each of the methods will be tested for a robotic team size of $n = 2, 3, 4$. All three environments found in Figure 4.1 are used in the different experiments. Each experiment is repeated five times for each method to get an accurate picture of the repeatability of the results. In Table 4.1 the parameters for each of the experiments in this section are detailed.

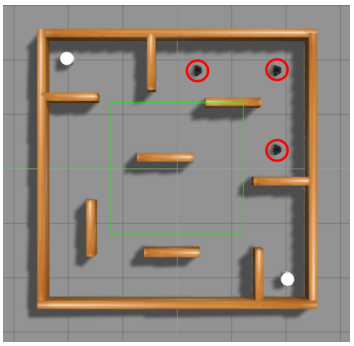
The starting positions for each of the benchmarking experiments are shown in Figure 4.2. To increase the visibility, the starting positions are circled in red.



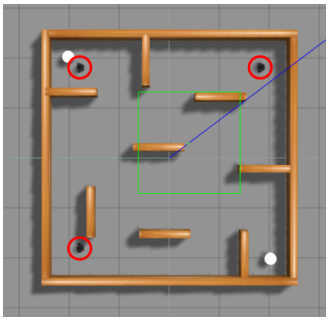
(a) Starting Positions Experiment 1



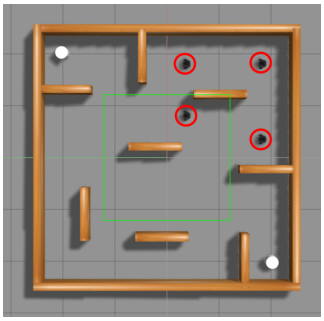
(b) Starting Positions Experiment 2



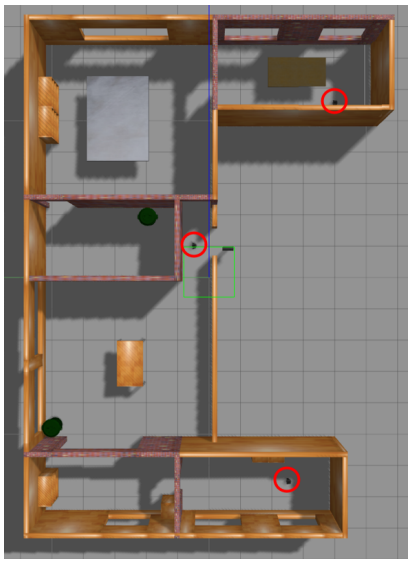
(c) Starting Positions Experiment 3



(d) Starting Positions Experiment 4



(e) Starting Positions Experiment 5



(f) Starting Positions Experiment 6

Figure 4.2: Benchmarking Tests: Starting Positions

Table 4.2: Scalability Experiment 1 Design Parameters

Experiment number	**Environment**	**Number of Robots**	**Starting Positions**
1.1a	TB3 Stage 4	2	far, Fig. 4.3(a)
1.1b	TB3 Stage 4	2	close, Fig. 4.3(b)
1.2a	TB3 Stage 4	3	far, Fig. 4.3(c)
1.2b	TB3 Stage 4	3	close, Fig. 4.3(d)
1.3a	TB3 Stage 4	4	far, Fig. 4.3(e)
1.3b	TB3 Stage 4	4	close, Fig. 4.3(f)
1.4a	TB3 Stage 4	5	far, Fig. 4.3(g)
1.4b	TB3 Stage 4	5	close, Fig. 4.3(h)

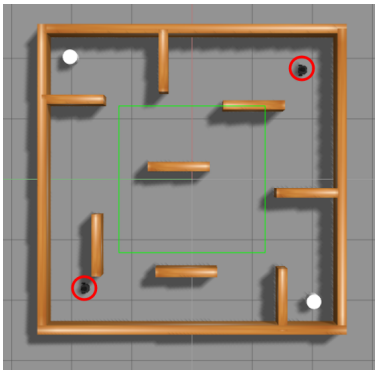
4.1.2.2 Scalability Tests

The scalability tests seek to determine how scalable the new k-means method is. These tests are only conducted for the k-means method. The performance of the method is considered over a large range of robotic team sizes, including robotic team sizes of $n = 2, 3, 4, 5, 7, 9$. For each of the robotic team sizes the experiment is repeated five times.

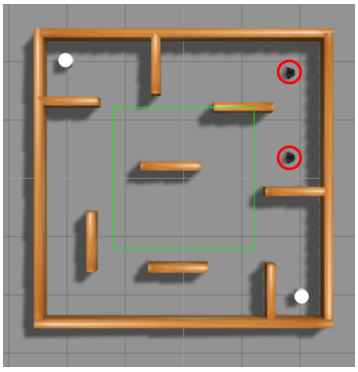
For the first set of experiments, the Stage 4 environment is used for four different sizes of robotic teams and two different starting positions: close and far. The TB3 Stage 4 environment is used in these experiments for robotic team sizes of $n = 2, 3, 4, 5$. The starting points are also varied as close and far for these experiments. This is done to isolate the effects of starting the robots close and far from each other on the completion time, distance travelled, and quality of the completed merged map. Table 4.2 outlines the parameters for the different experiments for Experiment 1.

The different starting positions and environments can be seen in Figure 4.3. To increase the visibility, the starting positions are circled in red.

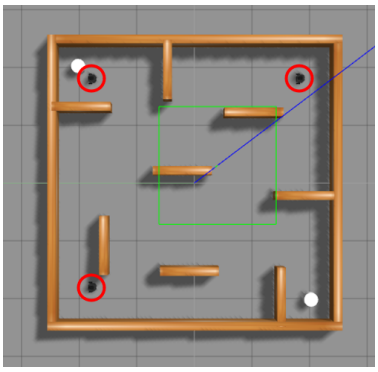
For the second set of experiments, the TB3 House environment is used for robot team sizes $n = 3, 5, 7, 9$. The second set of experiments is run so the scalability can be tested over a much larger environment and determine the effect that the robotic team size can have versus the size of the environment. Table 4.3 shows the parameters for the different experiments for Experiment 2.



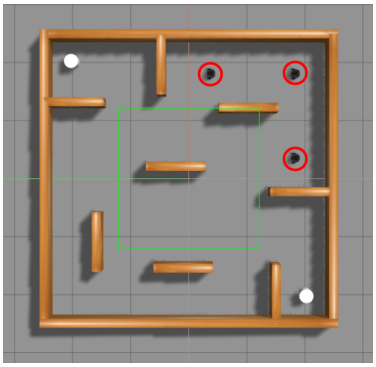
(a) Experiment 1.1a Far



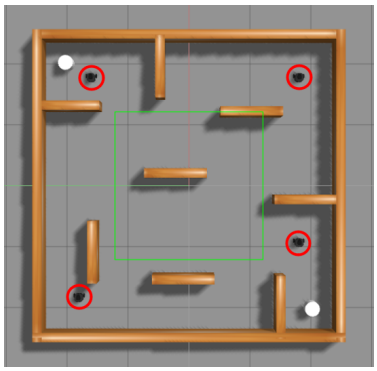
(b) Experiment 1.1b Close



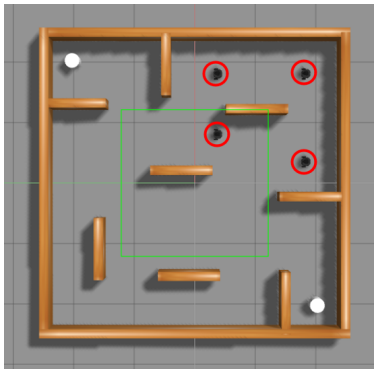
(c) Experiment 1.2a Far



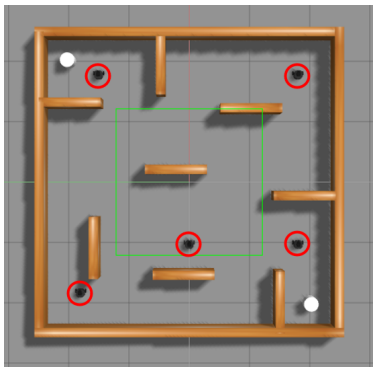
(d) Experiment 1.2b Close



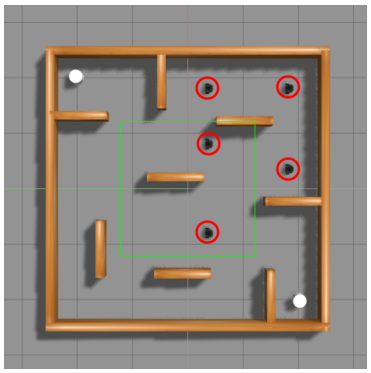
(e) Experiment 1.3a Far



(f) Experiment 1.3b Close



(g) Experiment 1.4a Far



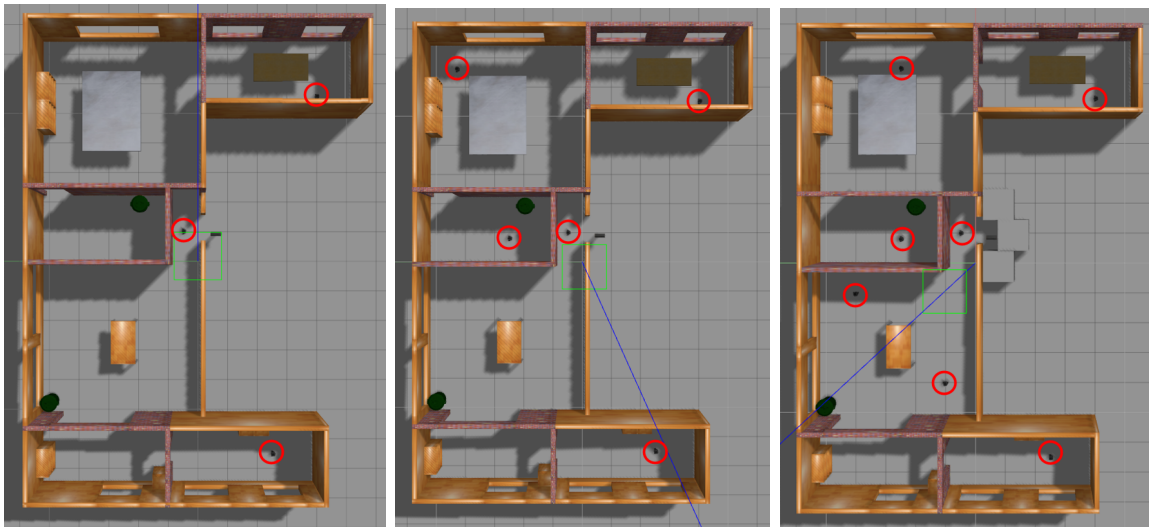
(h) Experiment 1.4b Close

Figure 4.3: Scalability Experiment 1: Starting Positions

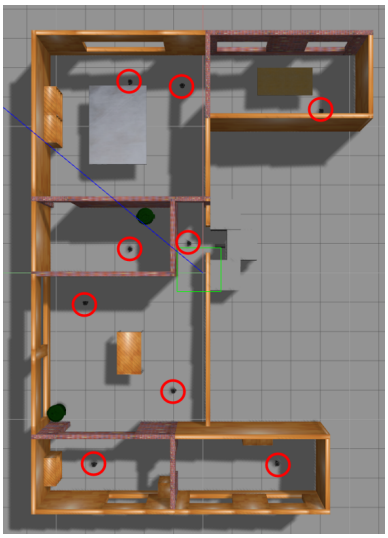
In Figure 4.4 the different starting points for the experiments are noted in the environment, starting points are circled in red.

Table 4.3: Scalability Experiment 2 Design Parameters

Experiment number	**Environment**	**Number of Robots**	**Starting Positions**
2.1	TB3 House	3	Fig. 4.4(a)
2.2	TB3 House	5	Fig. 4.4(b)
2.3	TB3 House	7	Fig. 4.4(c)
2.4	TB3 House	9	Fig. 4.4(d)



(a) Experiment 1 (b) Experiment 2 (c) Experiment 3



(d) Experiment 4

Figure 4.4: Scalability Experiment 2: Starting Positions

4.2 Real-World

A set of real-world tests and experiments are run to confirm the findings of the simulated results. The environment and different tests are detailed in the sections below.

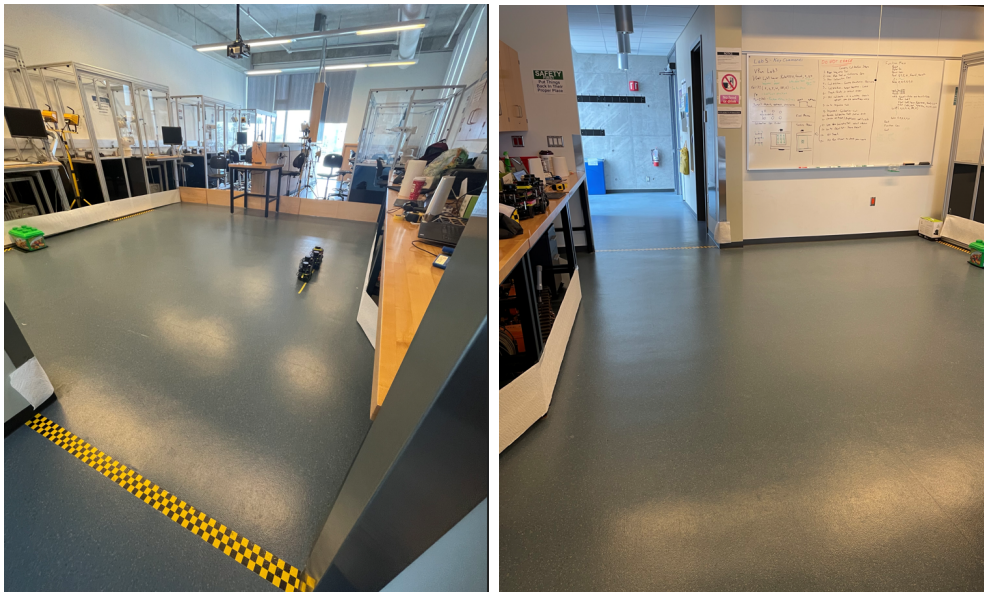
4.2.1 Real-World Environments

The real-world tests are carried out in two different locations, a hallway and lab environment. The hallway environment is shown in Figure 4.5, where a view of the left side and right side of the hall is shown. For these experiments full results are recorded in the main body of the thesis. The lab environment is shown in Figure 4.6,

(a) Hallway Left (b) Hallway Right

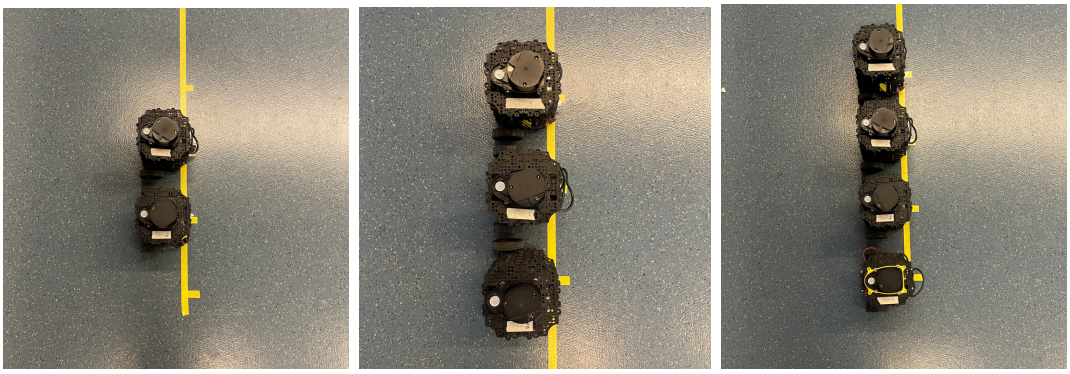
Figure 4.5: Real-World Environment: Hallway [37]

where a picture facing north and south in the environment is shown. The starting points for both of the environments were the same in relative location to each other. Each robot is placed 20 cm away from the other robots and the starting locations were determined in relative locations to the first robot. The starting points for the



(a) Lab Environment North (b) Lab Environment South

Figure 4.6: Real-World Environment: Lab



(a) 2 Robots (b) 3 Robots (c) 4 Robots

Figure 4.7: Real-World Tests: Starting Positions

experiments can be seen in Figure 4.7. For each of these tests the results are recorded for one run. The experiments are broken in two different sections based upon the environment used.

Table 4.4: Hallway Tests Design Parameters

Experiment number	**Environment**	**Number of Robots**	**Benchmark**	**Starting Positions**
1	Hallway	2	yes	Fig. 4.7(a)
2	Hallway	3	no	Fig. 4.7(b)
3	Hallway	4	no	Fig. 4.7(c)

4.2.2 Real-World Tests

The real-world tests are explained in the sections below starting with the hallway environments tests and moving onto the lab environment tests.

4.2.2.1 Hallway Tests

The preliminary tests are carried out in the hallway environment. The tests are detailed in Table 4.4. In this experiment some of the tests are benchmarked, meaning performed for all three of the methods: the no coordination, shared map, and k-means methods. The experiments were carried out for robotic team sizes $n = 2, 3, 4$. If the test is conducted for all three of the methods, it is recorded in the column labelled "Benchmark" this is shown as either "yes" or "no". Benchmarking was not carried out for all three robotic team sizes because the hallway environment proved to be too small for bigger sizes of robotic teams.
The relative starting locations can be seen in Figure 4.7. The placement in the environment can be seen in Figure 4.5(a).

4.2.2.2 Lab Tests

The tests carried out for the lab environment are detailed in Table 4.5. All of the test results for this environment were carried out for all three methods. The experiments were carried out for robotic team sizes $n = 2, 3, 4$.
The relative starting locations can be seen in Figure 4.7. The placement in the envi-

Table 4.5: Lab Tests Design Parameters

Experiment number	**Environment**	**Number of Robots**	**Benchmark**	**Starting Positions**
1	Lab	2	yes	Fig. 4.7(a)
2	Lab	3	yes	Fig. 4.7(b)
3	Lab	4	yes	Fig. 4.7(c)

ronment can be seen in Figure 4.6(a).

4.3 Summary

There are two main distinctions in the tests carried out for the validation of the developed k-means method: the simulated and real-world experiments.

For the simulated experiments two types of tests are performed, the benchmarking tests and scalability tests. The benchmarking tests are performed to determine the performance of the method versus two other methods. The diversity of these tests is maximized in the use of three different simulated environments, four different size robotic teams ($n = 2, 3, 4, 5$), and close and far starting locations. The scalability test seeks to test the limits of the number of robots the method can facilitate and the effect of far and close starting points on the method. These tests are performed over a robotic team sizes of $n = 2, 3, 4, 5, 7, 9$ and in two different environments to identify trends. Each of the experiments is repeated five times for all of these tests to determine the repeatability of the results obtained.

The real-world tests seek to validate some of the findings from the simulated experiments. This involves the use of two different environments, the use of three different robotic team sizes $n = 2, 3, 4$ and benchmarking with the two other methods. Only close starting points are considered for the real-world tests due to the increased complexity of close starting points on the system. Each of these experiments is performed once.

Chapter 5

Results and Discussion

In this chapter the results from the experiments as mentioned in Chapter 4 are recorded. Full results from all the individual runs can be found in Appendix A. First a review of the simulation results is presented. This is followed by a review of the real-world results.

5.1 Simulation Results

In this section a review of the different results obtained from the simulations are discussed, starting with the benchmarking experiments and then the results from the two scalability experiments.

5.1.1 Benchmarking Tests Results

The benchmarking experiments were completed for various robotic team sizes, in various environments, and using close and far starting locations. A review of the different results for the methods can be seen in the following tables and figures. For each experiment the average time, average distance and standard deviations are recorded.

Table 5.1: Benchmarking Experiment 1 Results

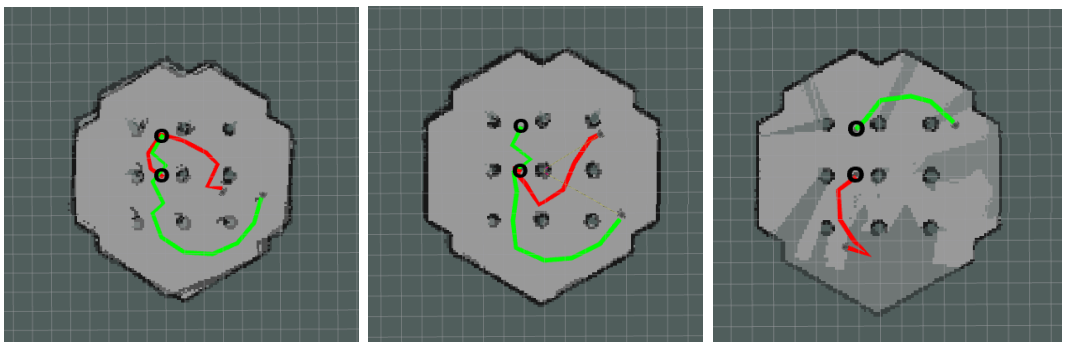
Measure	**No Coordination**	**Shared Map**	**K-means**
Avg. Time (s)	140.551	140.186	76.960
Std. Time (s)	31.112	27.627	7.462
Avg. Distance (m)	5.542	4.633	2.960
Std. Distance (m)	1.157	1.184	0.569

Figures of the paths taken and merged maps are also shown.

5.1.1.1 Benchmarking Experiment 1 Results

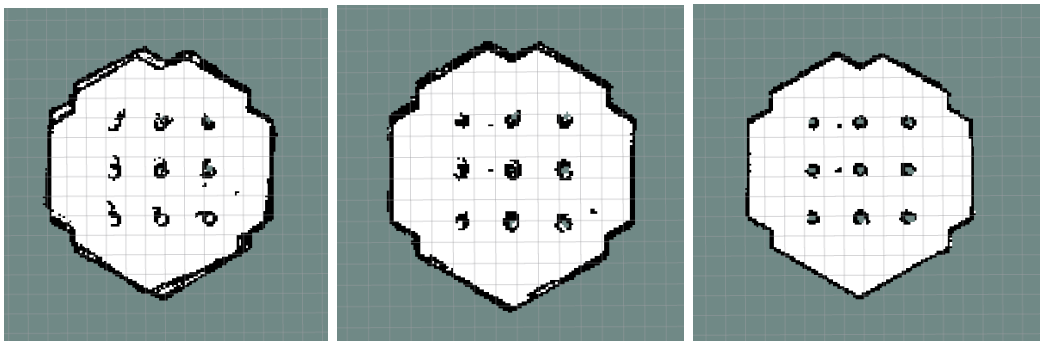
Experiment 1 involved a robotic team size of $n = 2$ robots. The results from this experiment are also published in previous work by Goodwin and Nokleby [37]. The experiments were carried out with all three methods as described in the experimental design section. The environment and starting points can be seen in Figure 4.2(a). The results can be seen in Table 5.1. From this table it can be seen that the k-means method outperforms the two other benchmark methods by far in average distance and time measures. The new method improves the exploration task with by reducing the exploration time by a minimum of 45.1% and a reducing the distance travelled by a minimum of 36.1% compared to the other methods. Additionally, the standard deviation for the k-means method in both time and distance are significantly smaller values that the other methods. This means the results for the k-means method are more repeatable and reliable.

The path comparisons can be seen in Figure 5.1. The different agents' paths are in different colours on the maps and the starting points are shown as black circles. It can be seen that using the k-means method provides a more logical path where one robot focuses on the top half of the map and the other focuses on the bottom half. It is also seen that the paths are much shorter than the other methods in comparison. Figure 5.2 shows the merged maps. As can be seen in the figure, the k-means method



(a) No Coordination (b) Shared Map (c) K-Means

Figure 5.1: Benchmarking Experiment 1: Path Comparisons [37]



(a) No Coordination (b) Shared Map (c) K-Means

Figure 5.2: Benchmarking Experiment 1: Merged Maps [37]

provides a much cleaner merged map. This is because less overlap in the map reduces the amount of error introduced by dead-reckoning.

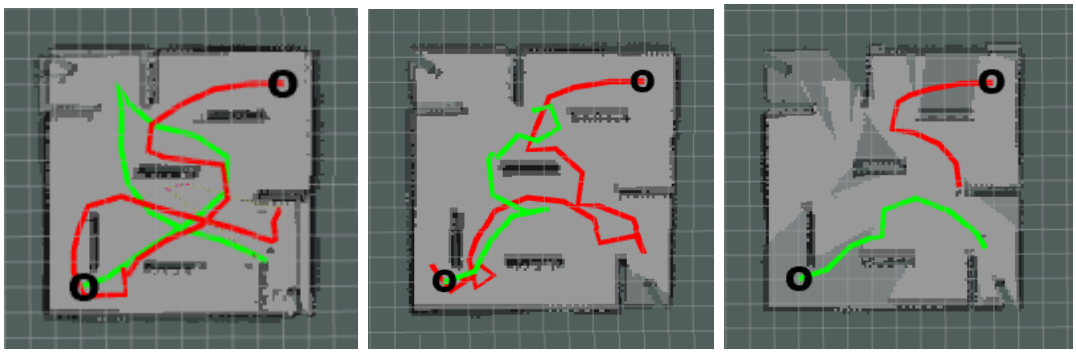
5.1.1.2 Benchmarking Experiment 2 Results

Experiment 2 was carried out again with a robotic team size of $n = 2$. However, the environment was changed and the agents were started farther away from each other as can be seen in Figure 4.2(b). The average results can be seen in Table 5.2. From the table it becomes clear again that the proposed k-means method outperforms both the other methods. The completion time was reduced by a minimum of 43.7% and the distance travelled was reduced by a minimum of 53.3% compared to the other methods.

Table 5.2: Benchmarking Experiment 2 Results

Measure	No Coordination	Shared Map	K-means
Avg. Time (s)	266.865	262.579	147.837
Std. Time (s)	94.090	46.021	28.946
Avg. Distance (m)	11.155	9.776	4.568
Std. Distance (m)	2.674	2.877	0.904

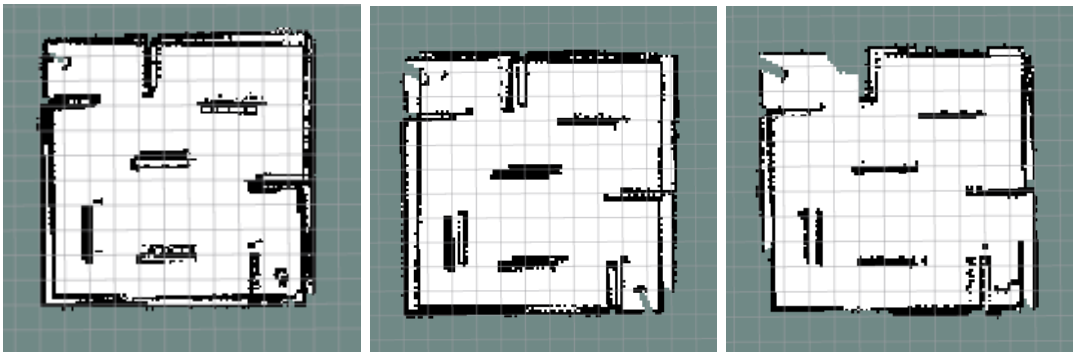
The different paths taken by one run for each of the methods can also be seen in Figure 5.3. It can be seen that there is a significant improvement in the paths taken



(a) No Coordination (b) Shared Map (c) K-Means

Figure 5.3: Benchmarking Experiment 2: Path Comparisons

by the k-means method. The k-means method promotes keeping the two robots in two separate halves of the map and the paths taken are much shorter by inspection. The merged map topics can be seen in Figure 5.4. It can be noted that starting the



(a) No Coordination (b) Shared Map (c) K-Means

Figure 5.4: Benchmarking Experiment 2: Merged Maps

robots farther away from each other seems to introduce more error in the merged

Table 5.3: Benchmarking Experiment 3 Results

Measure	**No Coordination**	**Shared Map**	**K-means**
Avg. Time (s)	425.300	535.700	174.477
Std. Time (s)	139.165	150.407	22.568
Avg. Distance (m)	11.775	10.377	3.778
Std. Distance (m)	2.879	4.122	1.288

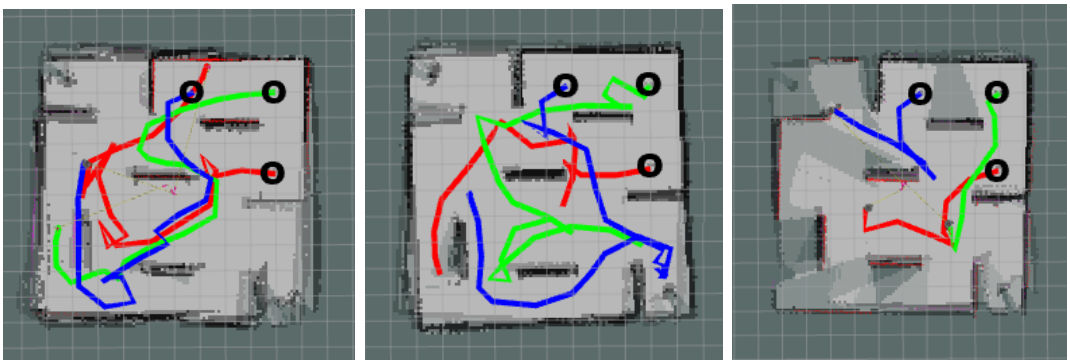
maps than starting them closer together. However, even with this considered there is a slight improvement in the quality of the merged map observed by the k-means method. Also, there is a small section missed in the top left corner of the k-means map, however this may be due to the explore_lite algorithm which only recognizes frontier regions of a certain threshold size.

5.1.1.3 Benchmarking Experiment 3 Results

For Experiment 3, the robotic team size was increased by 1 to a size of $n = 3$. The starting points and the environment can be seen in Figure 4.2(c). The average results can be seen in Table 5.3. It can be seen from the values in this table that the k-means method outperformed the other methods again with a minimum reduction in completion time of 59.0% and a minimum reduction of 63.6% in the distance travelled. The new method shows a significant improvement in the repeatability and reliability due to the small standard deviation values.

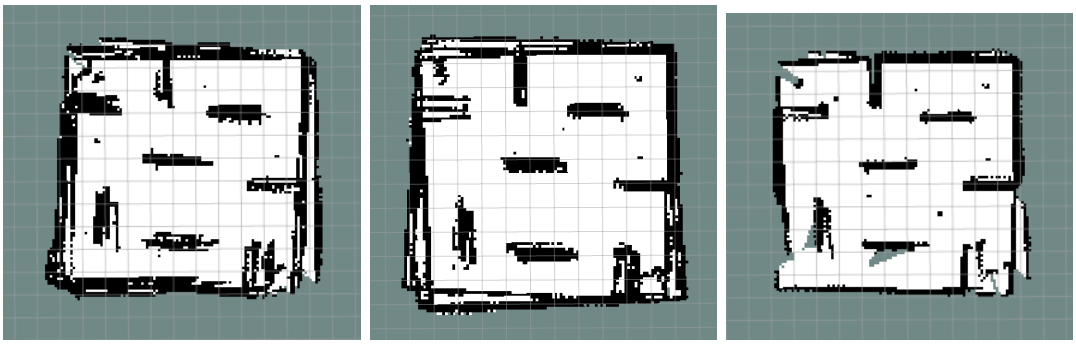
The paths taken for a run of each method are compared in Figure 5.5. As can be seen in the figure the paths taken by the k-means method are much shorter and provide better segmentation of the map. The other methods provide a lot of overlap in the paths taken.

The merged maps can be seen in Figure 5.6. From these maps it becomes clear that the k-means method provides a more accurate merged map. As in the previous experiment some pieces of the map are missing, again this may be due to the explore_lite



(a) No Coordination (b) Shared Map (c) K-Means

Figure 5.5: Benchmarking Experiment 3: Path Comparisons



(a) No Coordination (b) Shared Map (c) K-Means

Figure 5.6: Benchmarking Experiment 3: Merged Maps

algorithm setting the minimum threshold values for a frontier region.

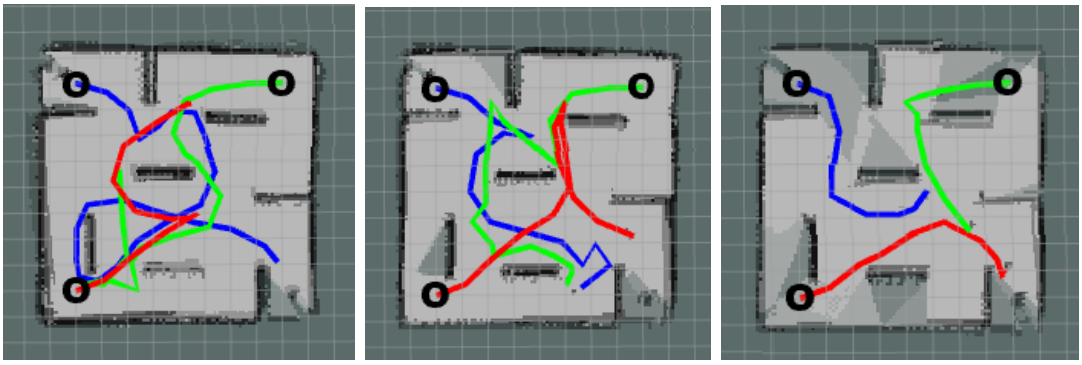
5.1.1.4 Benchmarking Experiment 4 Results

In Experiment 4 the same environment was used as in the two previous experiments. The starting points were farther away from each other as can be seen in Figure 4.2(d). The results for the experiment can be seen in Table 5.4. As can be seen in the table the k-means method outperforms the other two methods by far. The method reduces the completion time a minimum of 52.8% and reduces the distance travelled a minimum of 58.1% compared to the other methods.

The paths are compared for an average run of the simulation in Figure 5.7. As seen

Table 5.4: Benchmarking Experiment 4 Results

Measure	No Coordination	Shared Map	k-means
Avg. Time (s)	405.903	339.908	160.464
Std. Time (s)	136.791	131.804	35.572
Avg. Distance (m)	10.445	8.965	3.756
Std. Distance (m)	2.878	3.083	0.478

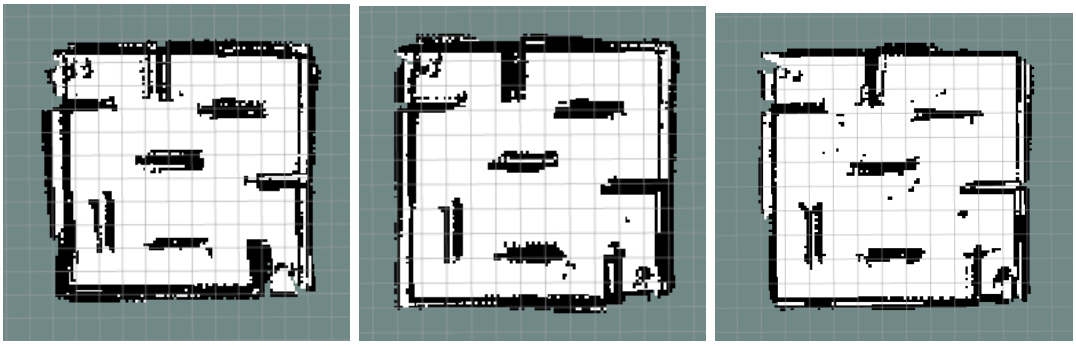


(a) No Coordination (b) Shared Map (c) K-Means

Figure 5.7: Benchmarking Experiment 4: Path Comparisons

in the figure the paths taken by the k-means method are much shorter and provide better segmentation of the map. There does not seem to be much improvement between the no coordination and the shared map methods.

In Figure 5.8 the merged map topics can be seen. There is quite an improvement in



(a) No Coordination (b) Shared Map (c) K-Means

Figure 5.8: Benchmarking Experiment 4: Merged Maps

the k-means method merged map as compared to the other methods. It can also be seen for the k-means method that the quality of the merged map is slightly worse

Table 5.5: Benchmarking Experiment 5 Results

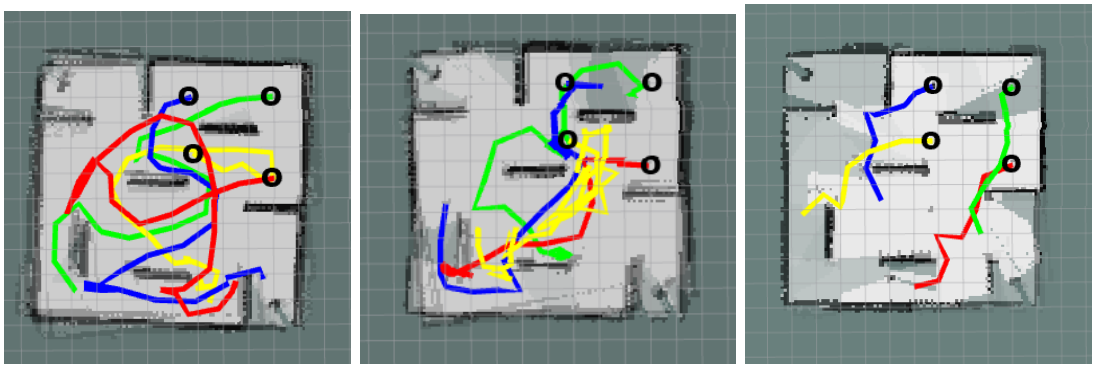
Measure	**No Coordination**	**Shared Map**	**K-means**
Avg. Time (s)	638.573	1288.532	295.895
Std. Time (s)	180.680	258.532	44.236
Avg. Distance (m)	11.155	11.013	4.226
Std. Distance (m)	2.821	4.445	1.463

than the Experiment 3 merged map in Figure 5.6. This is due to the increased error when starting the robots farther apart.

5.1.1.5 Benchmarking Experiment 5 Results

In Experiment 5 the number of robots is increased to $n = 4$. The environment is the same as the three previous experiments and the robots start off in close positions. The starting positions can be seen in Figure 4.2(e). In Table 5.5 the average results are recorded for the three different methods. It can be seen again that the k-means method far outperforms the other methods. The method outperforms the other methods with a minimum reduction in completion time of 53.7% and a minimum reduction in average distance travelled of 61.6%. The standard deviation for the k-means method is also much lower meaning the results are more repeatable than the other methods. A few things to note in these results, there is a spike in the time needed for the shared map method as compared to the other results. This is hypothesized to be due to the increase in the number of robots and the starting positions being so close. Also, the shared map method did not finish the exploration task two out of five runs.

The paths are compared in Figure 5.9. In these figures by inspection it is observed that the k-means method provides a much more well-organized exploration of the environment. It also provides much greater segmentation of the map, even with so many robots starting off close together. As mentioned previously the shared map

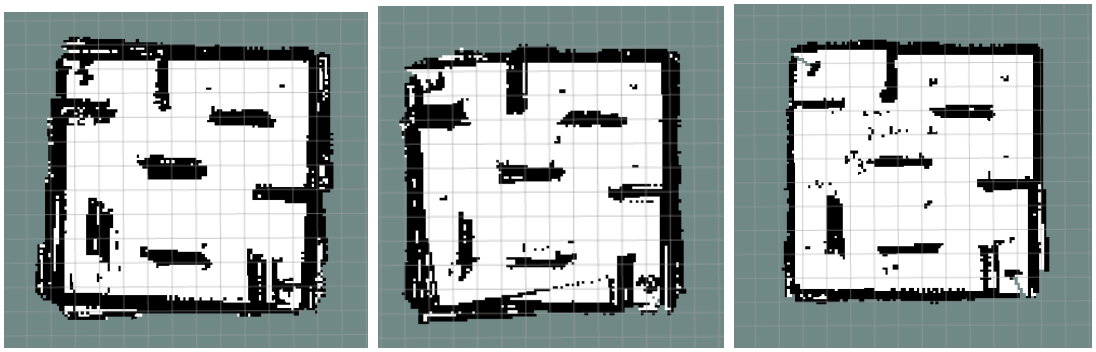


(a) No Coordination (b) Shared Map (c) K-Means

Figure 5.9: Benchmarking Experiment 5: Path Comparisons

method struggles with the coordination of four robots starting so close as can be seen in the paths taken.

The merged maps are presented in Figure 5.10. From the merged maps it can be seen



(a) No Coordination (b) Shared Map (c) K-Means

Figure 5.10: Benchmarking Experiment 5: Merged Maps

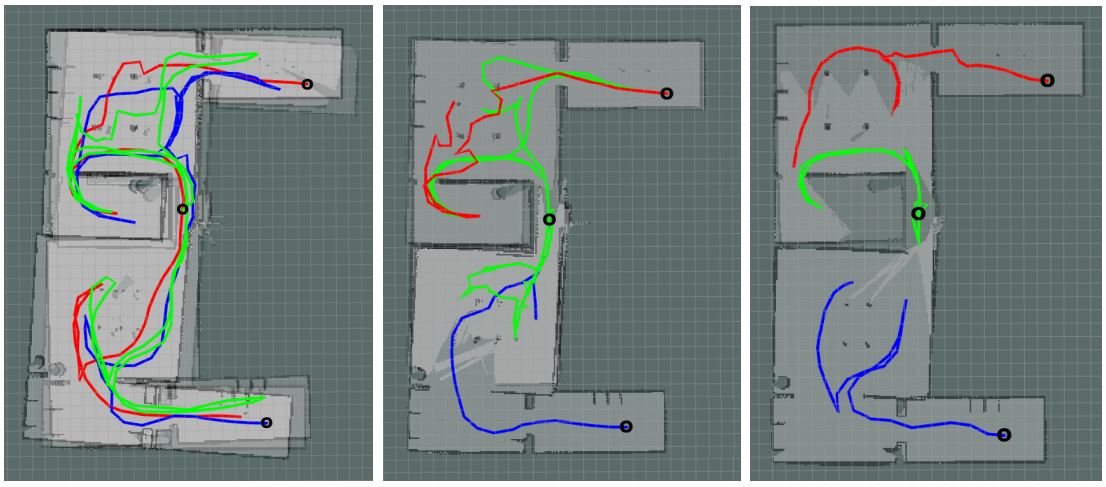
again that the accuracy of the map is improved using the k-means method. The no coordination merged map and the shared map methods provide very fuzzy borders for their maps. By inspection the quality of the k-means merged map is better than in Figure 5.8 due to the robots starting close together.

Table 5.6: Benchmarking Experiment 6 Results

Measure	No Coordination	Shared Map	K-means
Avg. Time	1515.485	1169.024	657.644
Std. Time	534.157	361.310	45.388
Avg. Distance	61.649	28.213	20.218
Std. Distance	15.931	16.044	3.797

5.1.1.6 Benchmarking Experiment 6 Results

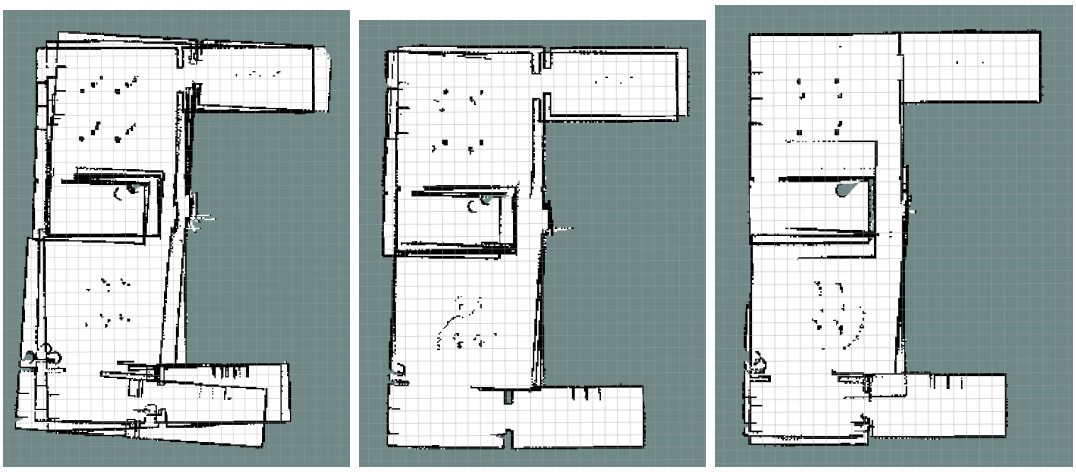
Experiment 6 was conducted in a much larger environment to test the algorithm in a more complex exploration task. The starting points for the experiment are shown in Figure 4.2(f). In Table 5.6 the average results for the experiments are shown. Again it can be seen that the results of the k-means method show a significant improvement over the other methods. The k-means method performs the exploration task with a minimum reduction of 43.7% in the completion time and a minimum reduction of 12.9% in the distance travelled compared to the other methods. The shared map method performed much more efficiently in this big environment with starting positions farther away, however, the method did not finish one of the five experiments. The path comparisons for Experiment 6 can be seen in Figure 5.11. From the paths



(a) No Coordination (b) Shared Map (c) K-Means

Figure 5.11: Benchmarking Experiment 6: Path Comparisons

taken by the three different methods it can be clearly seen that the k-means method takes a much more logical path for the experiment. The shared map method also performs much better than many of the previous experiments, further confirming the hypothesis that far starting positions is an easier task for the shared map method.
The merged map topics for Experiment 6 can be seen in Figure 5.12. The merged



(a) No Coordination (b) Shared Map (c) K-Means

Figure 5.12: Benchmarking Experiment 6: Merged Maps

map figures from this experiment are much less accurate than some of the previous experiments. This is believed to be due to how far the starting points are for this experiment. However, even with this taken into account, the k-means method does seem to marginally improve the merged map as compared to the shared map method and greatly improve the quality compared to the no coordination method.

5.1.2 Benchmarking Tests Discussion

For the benchmarking tests it becomes clear that the k-means method outperforms the no coordination and shared map methods in average time, distance travelled, paths taken and merged map accuracy.
In Figure 5.13 a graph of the different time results is shown to compare the different methods' performance over the experiments. The graph shows that the k-means

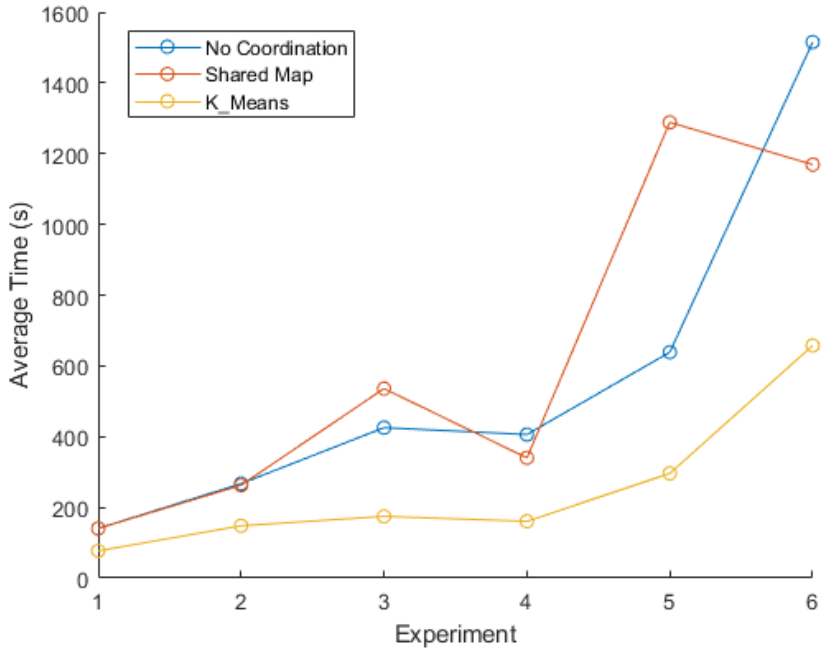


Figure 5.13: Benchmarking Experiments: Average Time Comparisons

method performs better for all of the experiments in completion time. An interesting note is that the shared map method did not always outperform the no coordination method. As noted before, the shared map method performs less efficiently when the agents start closer together as in Experiment 3 and 5.

However, to get a more accurate picture of the performance of the k-means method another graph is shown in Figure 5.14. In this figure the percentage improvement is shown for the k-means method. The graph shows that the k-means method improves the exploration task time by over 40% for all of the experiments and methods. It also shows that the k-means method tends to show a higher percentage improvement over the no coordination method for larger robotic teams. For the first two experiments with a robotic team size of $n = 2$ the k-means method shows a 45% improvement over both the methods. However, for robotic teams sizes of $n = 3, 4$ the percent improvement over the no coordination method jumps to a value of 55-60%. The k-means method percent improvement over the shared map method varies greatly for the last four experiments. This is due to the shared map method performing more efficiently for far starting position experiments.

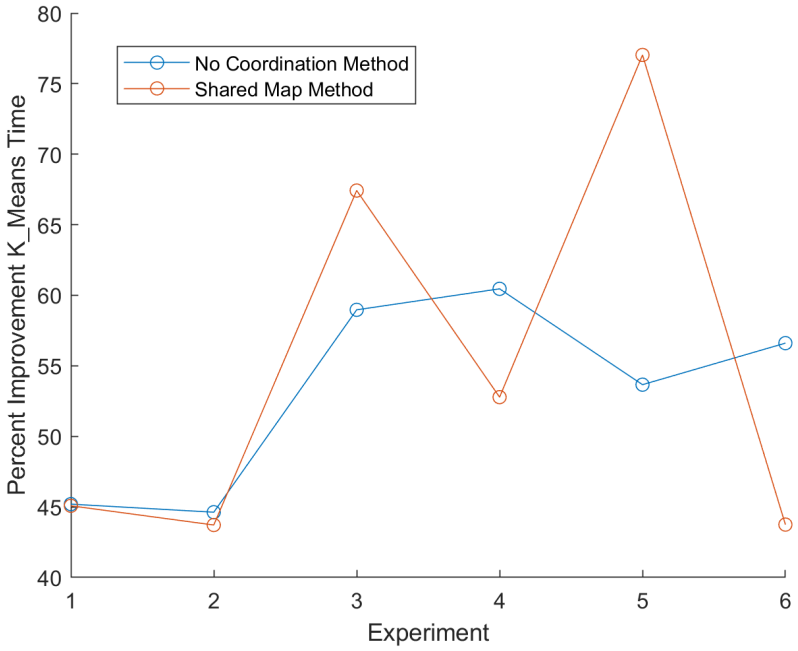


Figure 5.14: Benchmarking Experiments: Percent Improvement K-means Time

The k-means method also outperformed the other methods in average distance travelled by each of the agents. In Figure 5.15 the average distances for each of the experiments are plotted. From this figure it is easy to see that the k-means method decreases the distance travelled by each agent substantially on every experiment. The shared map method and no coordination method are similar in distance travelled by the individual agents except for in Experiment 6. This is because the map is much larger and the starting points are farther away. In Experiment 6 the shared map method performed much better than the no coordination method. Again to further show the improvement of the k-means method a percent improvement graph is plotted in Figure 5.16 for the average distances. From this figure the k-means method can also be seen to show significant improvement over the distances for both methods. For the no coordination method the k-means method reduced the distance travelled by over 45% for all the experiments and showed around 60-70% improvement for 5 out of 6 of the experiments. For the shared map method the k-means method improvement by over 27% for all the experiments and showed a 50-65% improvement for 4 out of 6 of the experiments.

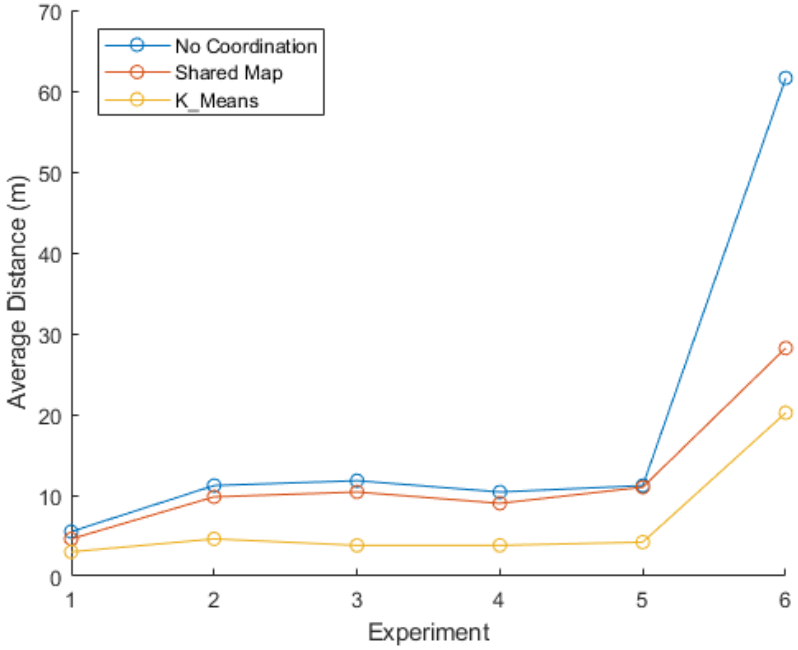


Figure 5.15: Benchmarking Experiments: Average Distance Comparison

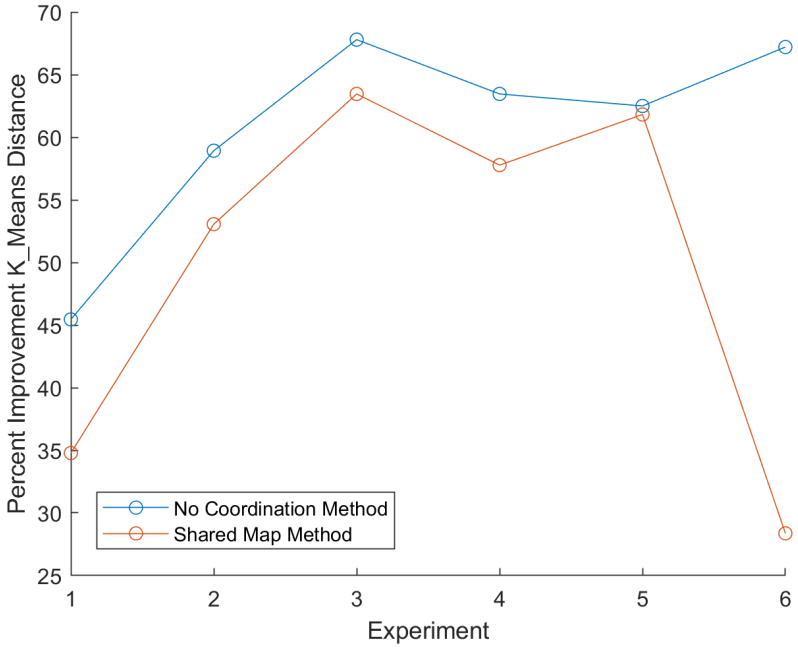


Figure 5.16: Benchmarking Experiments: Percent Improvement K-means Distance

The standard deviation of the time and distance results were also considered, as the standard deviation shows the predictability and repeatability of the results obtained.

In Figures 5.17 and 5.18 both the standard deviation for the time and distance results are shown respectively. In these figures the trends show that the k-means method

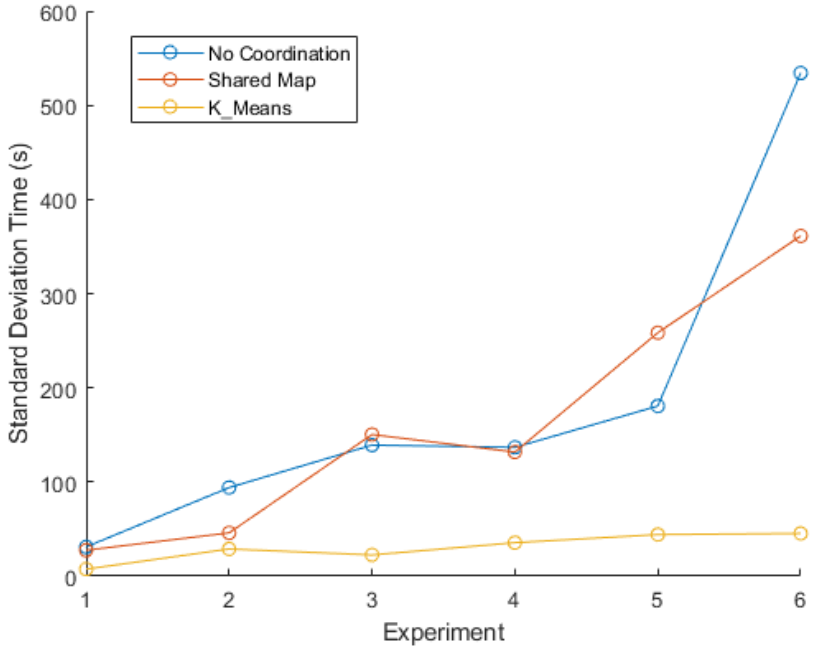


Figure 5.17: Benchmarking Experiments: Standard Deviation Time

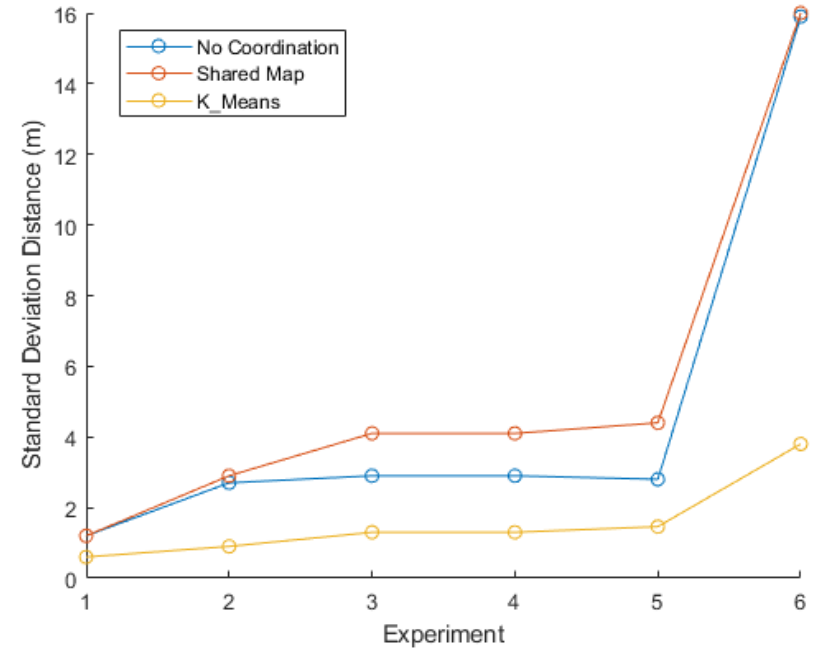
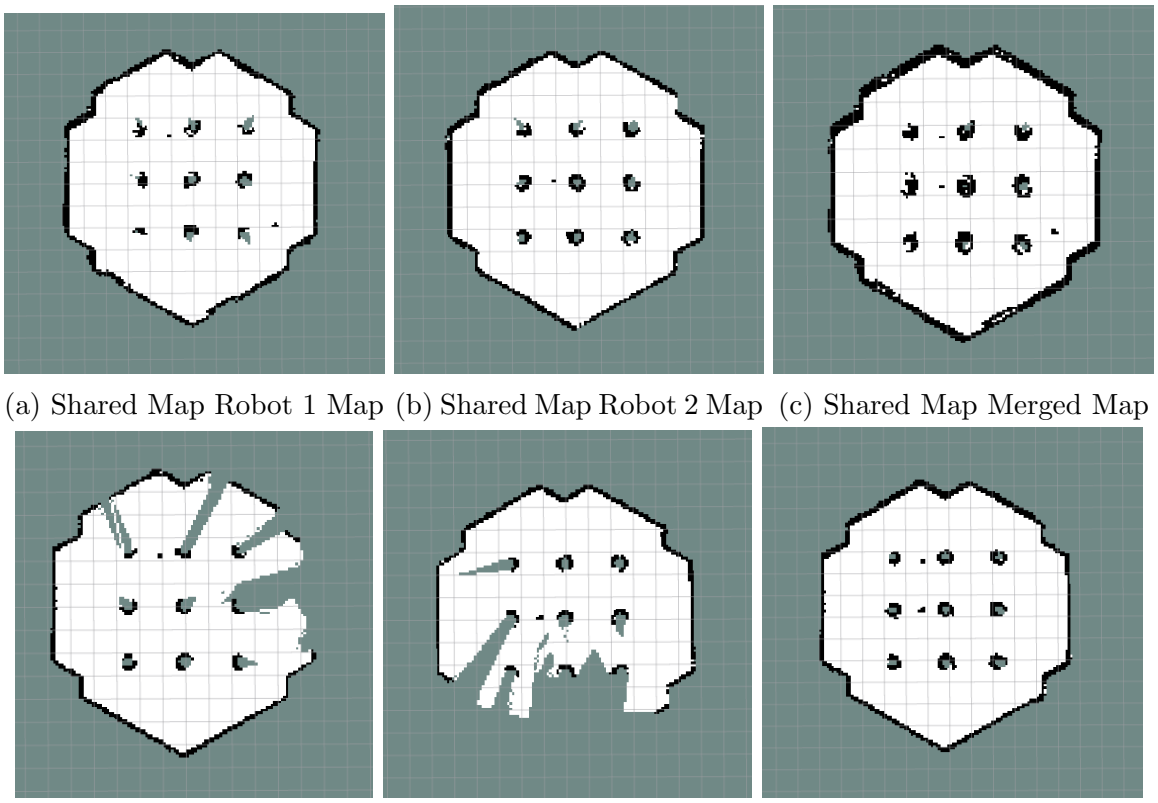


Figure 5.18: Benchmarking Experiments: Standard Deviation Distance

display a much smaller standard deviation in both time and distance measures than the two other methods presented. The two other methods also seem to have a positive correlation (an increase in the standard deviation) with the size of the environment; as the size of the environment is small in Experiment 1, medium in Experiments 2-5, and large in Experiment 6. The k-means method does not present much of a difference in standard deviation of the time results for all six experiments. Though the standard deviation for the distance results shows a small spike when the environment is changed to the biggest environment in Experiment 6.

Another trend to note in these experiments is the overlap in map area coverage. In Figure 5.19 a comparison of the merged maps from Experiment 1 of the shared map and k-means method is shown. By inspection it can be seen that the k-means



(a) Shared Map Robot 1 Map (b) Shared Map Robot 2 Map (c) Shared Map Merged Map

(d) K-means Robot 1 Map (e) K-means Robot 2 Map (f) K-means Merged Map

Figure 5.19: Benchmarking Experiment 1: Map Coverage Comparisons [37]

method decreases the overlap in map area covered. In the shared map method, the individual robots created two full maps even though they were sharing map informa-

tion. So, it can be seen that the k-means algorithm for task assignment decreases the overlap in area covered. This decrease in overlap of map area covered also seems to improve the quality of the merged map topic as seen in Figures 5.19(c) and 5.19(f).

5.1.3 Scalability Tests Results

In this section a review of the different experiments carried out to show the scalability of the system and determine the effect of differing starting points is reviewed. This sections' results correspond to Tables 4.2 and 4.3 found in Chapter 4.

5.1.3.1 Scalability Experiment 1 Results

For Experiment 1 the TB3 Stage 4 environment was used in Gazebo, as seen in Figure 4.1(b). The different starting points for the experiments can be seen in Figure 4.3. Experiment 1 is performed for both close and far starting points for robotic teams sizes of $n = 2, 3, 4, 5$.

The results for scalability Experiment 1 can be seen in Tables 5.7 and 5.8, where the experiment number, number of robots, average time (avg. time), standard deviation time (std. time), average distance (avg. dist.), and standard deviation distance (std. dist.) are recorded. In the results section the results are presented in two tables, the first table for far starting points, the second table for close starting points.

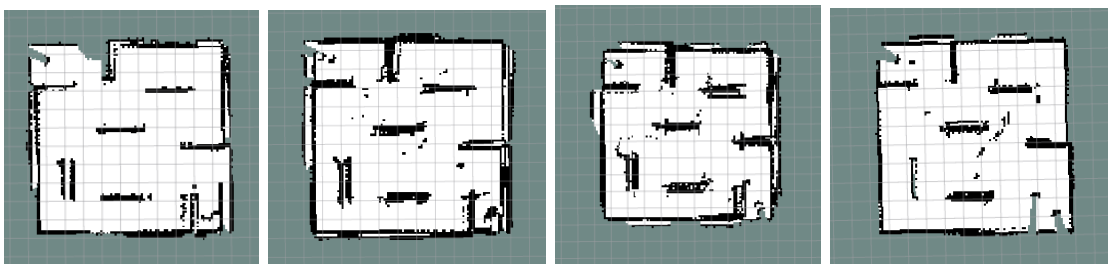
In Table 5.7, the results for scalability Experiment 1a for far starting points can be seen. For far starting points it can be seen that the average time increases as the robotic team size increases. This is likely due to the time it takes for the algorithm and simulation to run on the main computer. Additionally, the standard deviation also has a slightly positive correlation with the size of the robotic team, which indicates as the robotic team size increases the results are less predictable. However,

Table 5.7: Scalability Experiment 1 Results: Far Starting Points

Experiment Number	**Number of Robots**	**Avg. Time (s)**	**Std. Time (s)**	**Avg. Dist. (m)**	**Std. Dist. (m)**
1.1a	2	147.837	28.946	4.568	0.904
1.2a	3	160.464	35.572	3.921	0.345
1.3a	4	195.441	46.493	2.859	1.106
1.4a	5	303.777	41.767	2.346	1.100

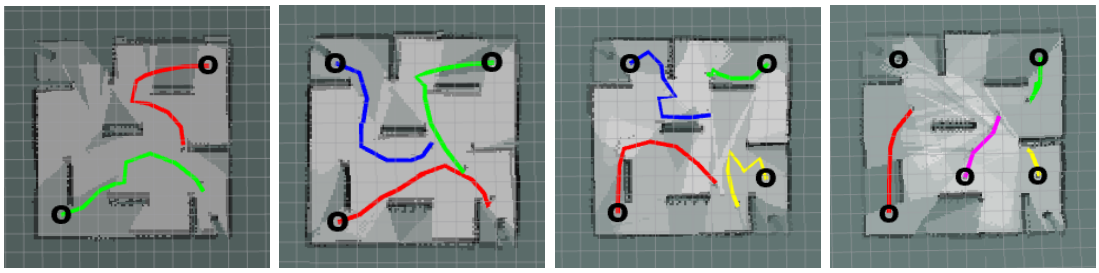
the average distance travelled by each agent steadily decreases with the increase in robotic team size, but again the standard deviation increases as the robotic team size increases.

The merged map topics are shown side-by-side for the four experiments starting far apart in Figure 5.20. There are no significant changes seen in the quality of the



(a) 2 Robots (b) 3 Robots (c) 4 Robots (d) 5 Robots

Figure 5.20: Scalability Experiment 1a: Merged Maps for Far Starting Points



(a) 2 Robot Paths (b) 3 Robot Paths (c) 4 Robot Paths (d) 5 Robot Paths

Figure 5.21: Scalability Experiment 1a: Paths for Far Starting Points

maps produced by the four different robotic team sizes, but the maps created by a robotic team size of $n = 2$ and $n = 5$ are by inspection the sharpest.

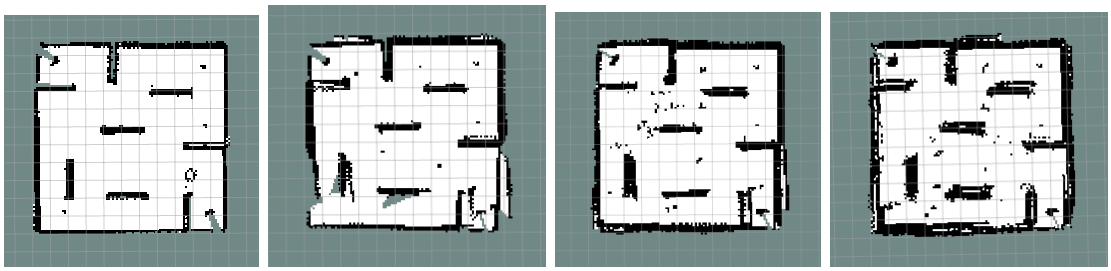
The paths taken by each of the agents are shown for all of the far starting experiments

Table 5.8: Scalability Experiment 1 Results: Close Starting Points

Experiment Number	Number of Robots	Avg. Time (s)	Std. Time (s)	Avg. Dist. (m)	Std. Dist. (m)
1.1b	2	216.920	19.240	7.523	1.374
1.2b	3	174.477	22.568	3.777	1.359
1.3b	4	295.895	44.236	4.752	1.533
1.4b	5	309.678	96.259	4.073	1.473

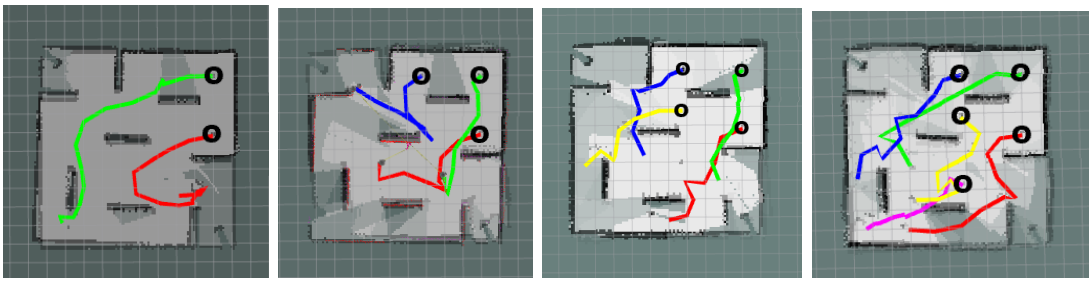
in Figure 5.21. The paths taken by each of the agents by inspection are logical and provide a good segmentation of the map. For a robotic team size of $n = 5$, only four of the five agents moved during the exploration because the robotic team size became too large compared to the size of the environment.

In Table 5.8, the results for scalability Experiment 1b, close starting points can be seen. For close starting points the average time decreased from $n = 2$ to $n = 3$ robots, but then increased as the size of the robotic team increased. The standard deviation,



(a) 2 Robots (b) 3 Robots (c) 4 Robots (d) 5 Robots

Figure 5.22: Scalability Experiment 1b: Merged Maps for Close Starting Points



(a) 2 Robot Paths (b) 3 Robot Paths (c) 4 Robot Paths (d) 5 Robot Paths

Figure 5.23: Scalability Experiment 1b: Paths for Close Starting Points

however, steadily increased with the size of the robotic team. The average distance

Table 5.9: Scalability Experiment 2 Results

Experiment Number	**Number of Robots**	**Avg. Time (s)**	**Std. Time (s)**	**Avg. Dist. (m)**	**Std. Dist. (m)**
1	3	657.644	45.388	19.932	3.999
2	5	737.805	148.070	13.119	2.923
3	7	744.856	187.992	9.035	2.476
4	9	743.499	169.701	5.234	2.037

travelled displayed more of a wave pattern. The standard deviation of the distances travelled remained fairly constant through all the experiments.

The merged map topics for the four different experiments can be seen in Figure 5.22. The merged map topics by inspection degrade in quality as the number of robots in the exploration increase. However, the merged map topic for robotic team size of $n = 2$ provides a seamless merged map.

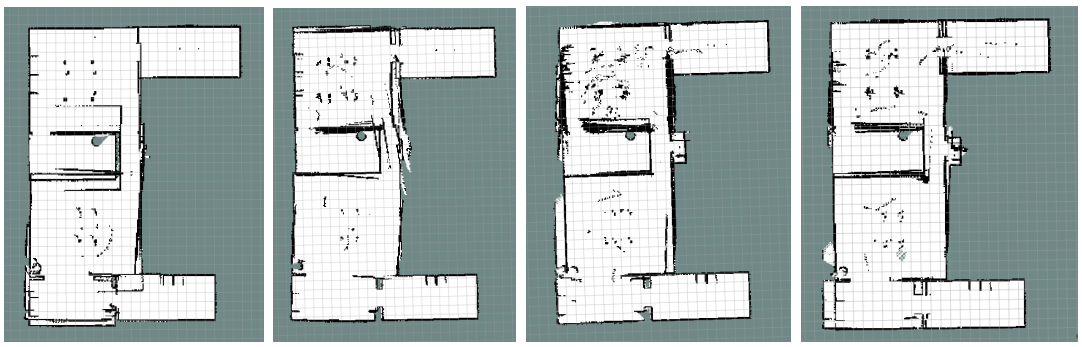
The paths taken by each of the agents is shown for all of the close starting experiments in Figure 5.23. The paths in the figure still provide good segmentation of the map, even though this is a much more difficult task than when starting from farther apart.

5.1.3.2 Scalability Experiment 2 Results

For scalability Experiment 2 the larger Gazebo environment was used, as seen in Figure 4.1(c). This test was performed over robotic team sizes of $n = 3, 5, 7, 9$. The results for Experiment 2 are found in Table 5.9. By inspection it can be seen that the average time and standard deviation of the time for the experiments increases from $n = 2$ to $n = 3$ robots, but remains fairly constant for robotic team sizes of $n = 5, 7, 9$. The average distance travelled for each agent decreases significantly with the increase in robotic team size. The standard deviation of the distance travelled also slightly decreases with the robotic team size.

The merged map topics for one of the runs for each experiment can be seen in Figure

5.24. From the merged map topics it can be seen that merging maps from far starting

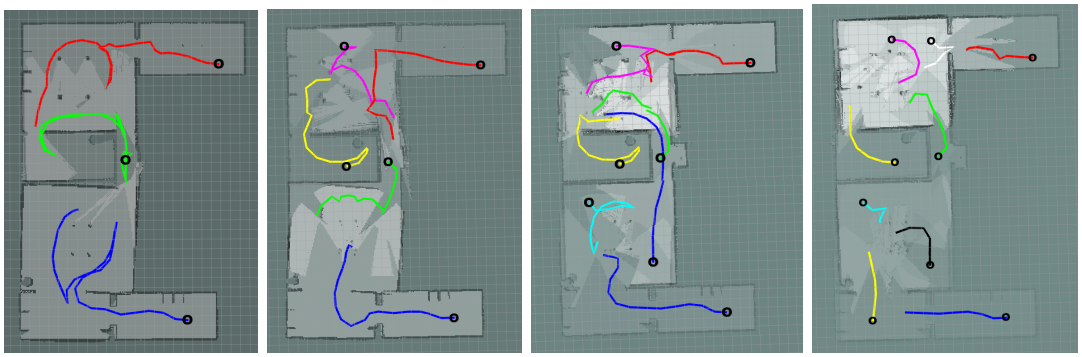


(a) 3 Robots (b) 5 Robots (c) 7 Robots (d) 9 Robots

Figure 5.24: Scalability Experiment 2: Merged Maps

points can be a difficult task. By inspection the quality of the maps does not seem to be much different for the four different experiments.

The paths from the four different experiments are shown in Figure 5.25. From the



(a) 3 Robot Paths (b) 5 Robot Paths (c) 7 Robot Paths (d) 9 Robot Paths

Figure 5.25: Scalability Experiment 2: Paths

paths taken it can be seen for the robotic team size of $n = 3$ the map is accurately separated into three horizontal sections. The challenge of segmenting the environment is greatly increased as the size in robotic team increases, however, the last three experiments still provide little to no overlap in the paths taken by each of the individual agents.

5.1.4 Scalability Tests Discussion

In this section a discussion and comparison of some of the results received for scalability Experiment 1 and Experiment 2 are detailed.

5.1.4.1 Scalability Experiment 1 Discussion

For scalability Experiment 1 the average time, average distance, and standard deviations of both time and distance are plotted for close and far starting points. This is to determine the effect of starting position on the performance of the developed k-means method.

In Figure 5.26 an average time comparison for Experiment 1 with close and far starting points is shown. From this figure, it can be seen by inspection than starting the

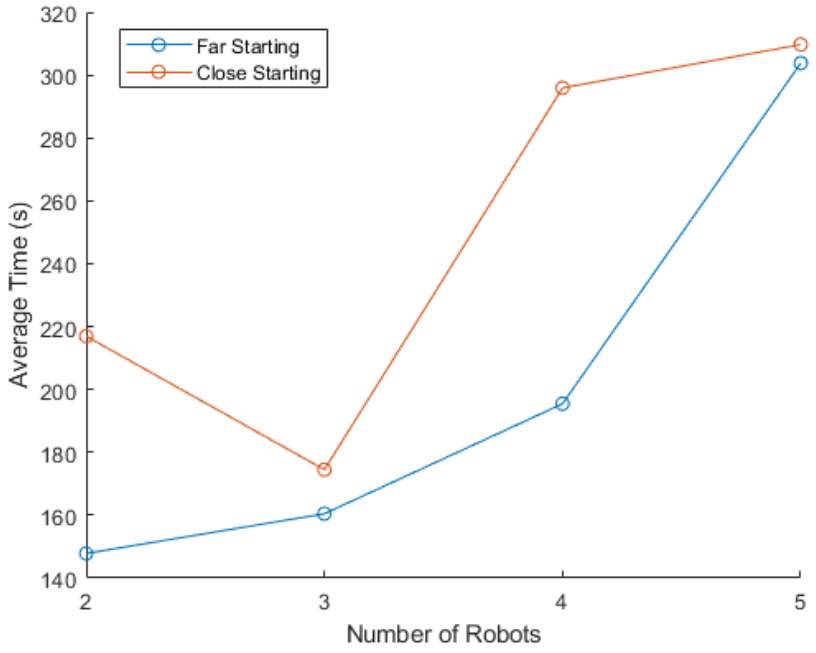


Figure 5.26: Scalability Experiment 1: Average Time Comparison

robots close together is more challenging that starting the robots farther apart in

terms of average exploration time. However, the trend with far starting points shows that the increase in robotic team size seems to increase almost exponentially. For close starting points, there is a dip in the graph initially from robotic team size of $n = 2$ to $n = 3$. Both close and far starting points tend to perform less efficiently with robotic team size of greater than $n = 3$. This is possibly due to the size of the environment being too small for a robotic team size of greater than $n = 3$, because increasing the robotic team size over this number makes the environment too crowded with obstacles.

Figure 5.27 shows the average distance travelled for each of the agents versus the number of robots for both close and far starting points. Again it can be concluded

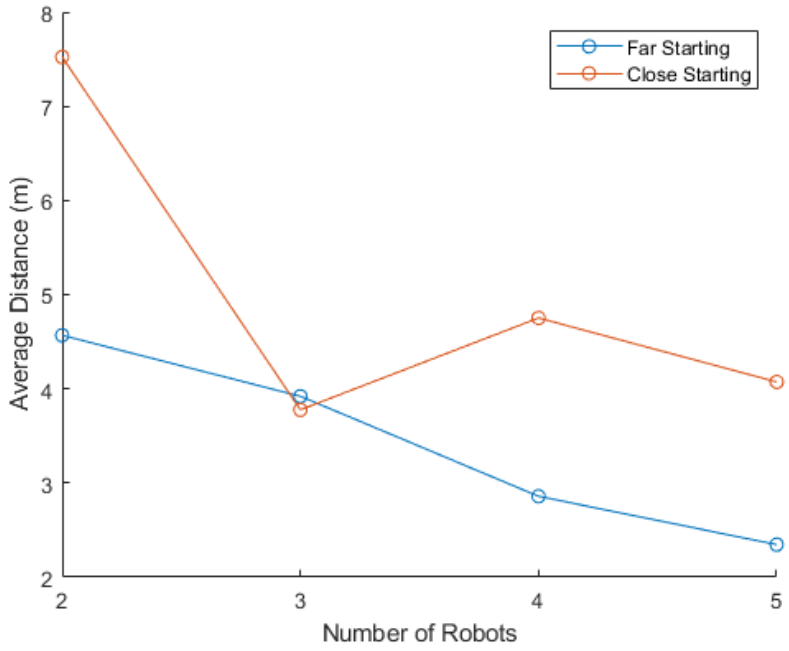


Figure 5.27: Scalability Experiment 1: Average Distance Comparison

that starting the robots far from each other provides an easier exploration task in both terms of time elapsed as well as distance travelled. The average distance travelled for the far starting experiments provides a linear decrease as the robotic team size increases. The close starting points, however, show a large decrease in distance travelled with a robotic team size of $n = 3$.

The standard deviation for the time is plotted in Figure 5.32. The standard devia-

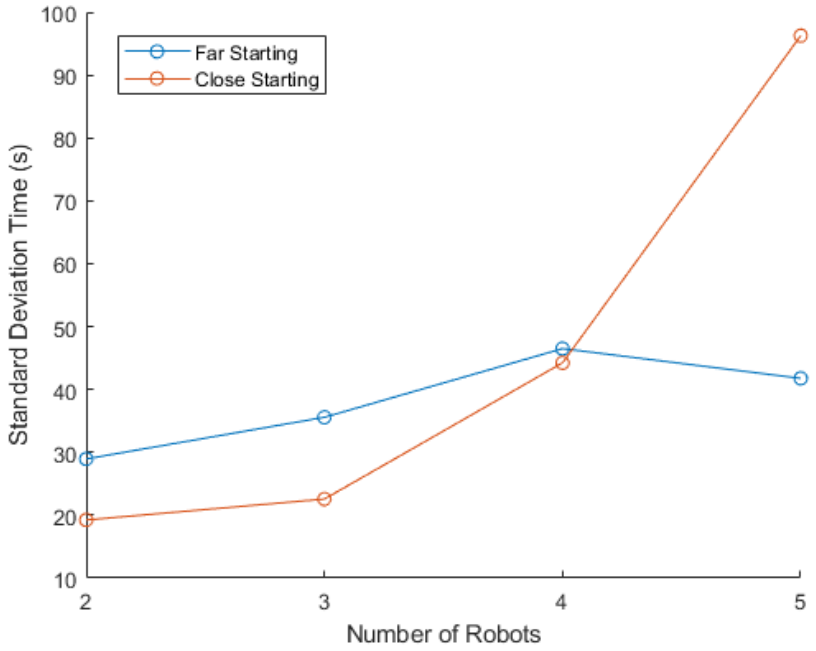


Figure 5.28: Scalability Experiment 1: Standard Deviation Time Comparison

tion for close starting experiments seems to grow almost exponentially on the graph. For the far starting experiments, the standard deviation for the completion time is fairly consistent. The standard deviation in the completion time for the close starting experiments start off with a lower value than for the far starting experiments, but surpasses the far starting experiments at the robotic team size of $n = 4$.

The standard deviation for the distance is plotted in Figure 5.29. For close starting points the standard deviation of the distance travelled stays fairly consistent for the four robotic team sizes. When considering the far starting points the standard deviation of the distance travelled decreases greatly when the robotic team size is $n = 3$ and remains close to a value of 1 for the remaining experiments. It is also seen that the standard deviation of the distance travelled for far starting points is much lower than for close starting points.

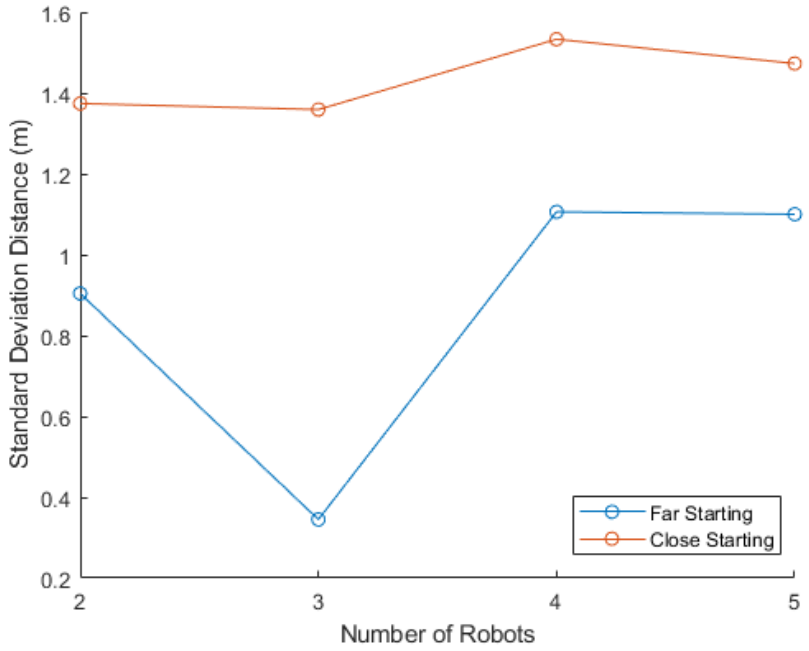


Figure 5.29: Scalability Experiment 1: Standard Deviation Distance Comparison

5.1.4.2 Scalability Experiment 2 Discussion

For scalability Experiment 2 the average time, average distance, and standard deviations of both time and distance are plotted to determine the effect of the size of the robotic team on the performance of the developed k-means method.

The average time was trended on a plot shown in Figure 5.30. It can be seen that the average time takes a large jump from a robotic team size of $n = 3$ to $n = 5$. Though the average time seems to plateau after the initial jump.

The average distance for the different experiments is also plotted in Figure 5.31. The average distance travelled by each agent for scalability Experiment 2 seems to have a fairly linear slope and a negative correlation to the number of robots in the team. This is a similar trend to scalability Experiment 1 for far starting points as observed in Figure 5.27. This is also confirmed by inspection when viewing Figure 5.25 where the average agents path length tends to decrease in size as the number of agents in the robotic team increases.

The standard deviation of the time is also plotted for Experiment 2 in Figure 5.32.

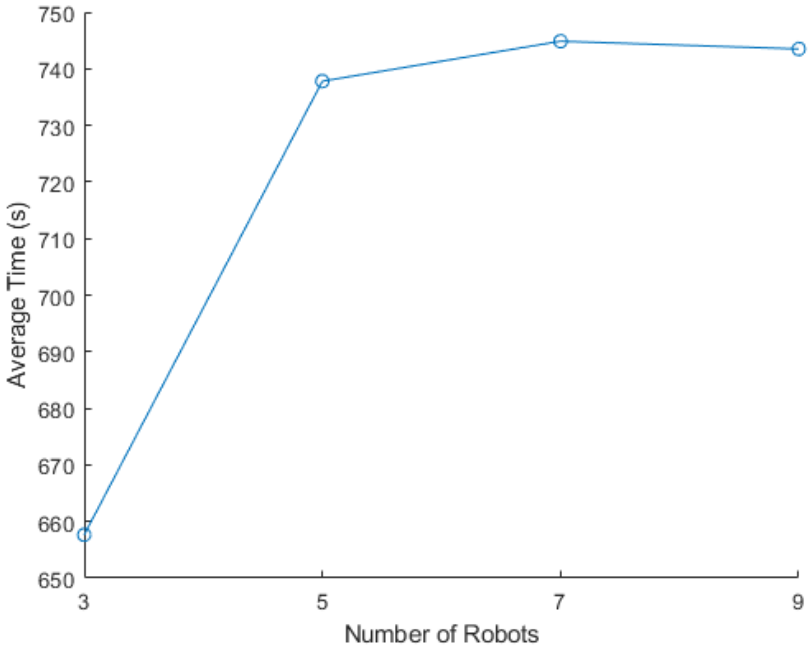


Figure 5.30: Scalability Experiment 2: Average Time

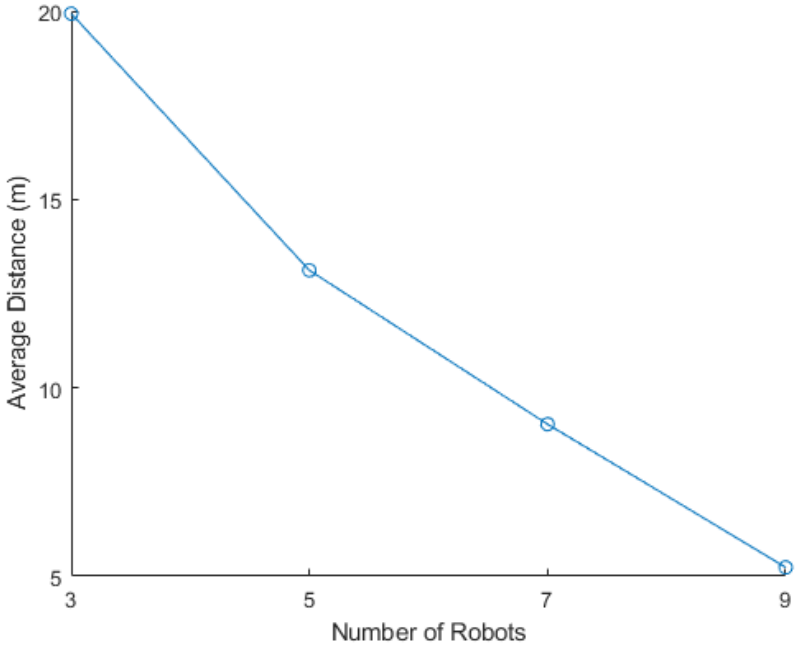


Figure 5.31: Scalability Experiment 2: Average Distance

Experiment 2 displays a similar trend in the standard deviation of the time to the trend shown in Figure 5.30 for the average time. This means the results become less

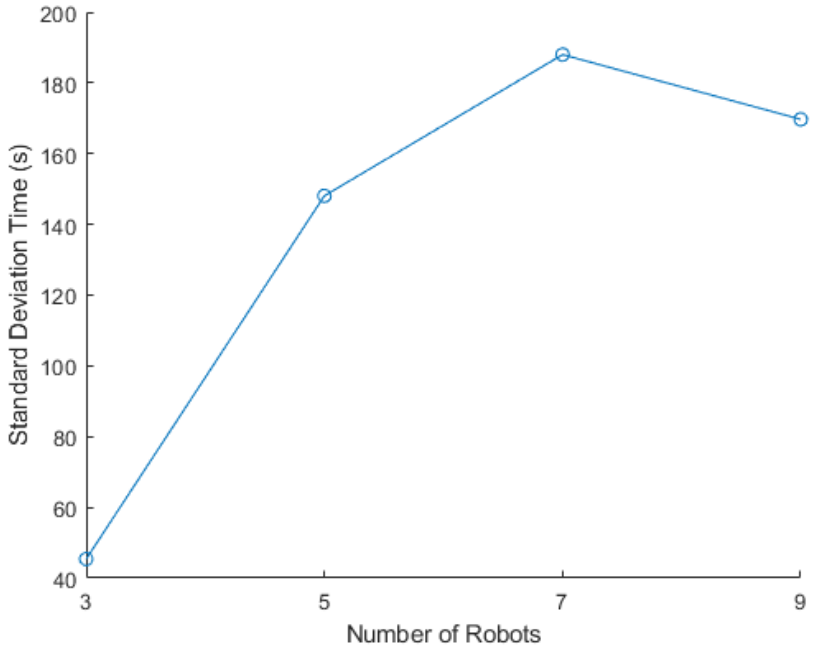


Figure 5.32: Scalability Experiment 2: Standard Deviation Time

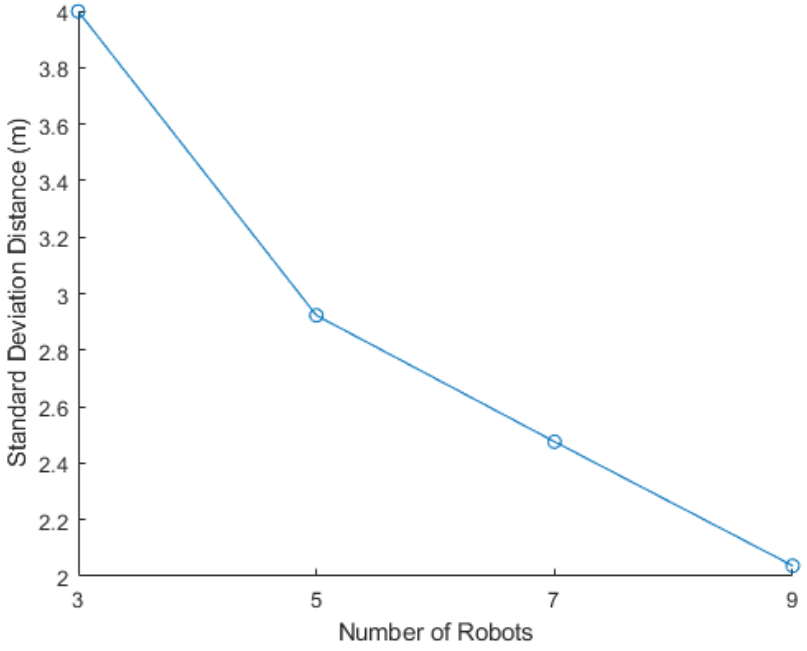


Figure 5.33: Scalability Experiment 2: Standard Deviation Distance

reliable after the robotic team size increases to $n = 5$, but there is no significant change for robotic team sizes of $n = 5, 7, 9$.

Table 5.10: Hallway Tests Results

Experiment Number	**Number of Robots**	**Method**	**Time (s)**	**Avg. Dist. (m)**
1	2	no coordination	151.556	22.764
1	2	shared map	190.939	23.686
1	2	k-means	83.525	10.120
2	3	k-means	98.850	9.822
3	4	k-means	114.655	14.689

Lastly, the standard deviation of the distance travelled is also plotted for the scalability Experiment 2 in Figure 5.33. Similarly to the average distance measures seen in Figure 5.31, the standard deviation of the distance travelled also decreases as the robotic team size increases. This means that with a higher size robotic team the average distance travelled will be more repeatable than with a smaller size robotic team.

5.2 Real-World Results

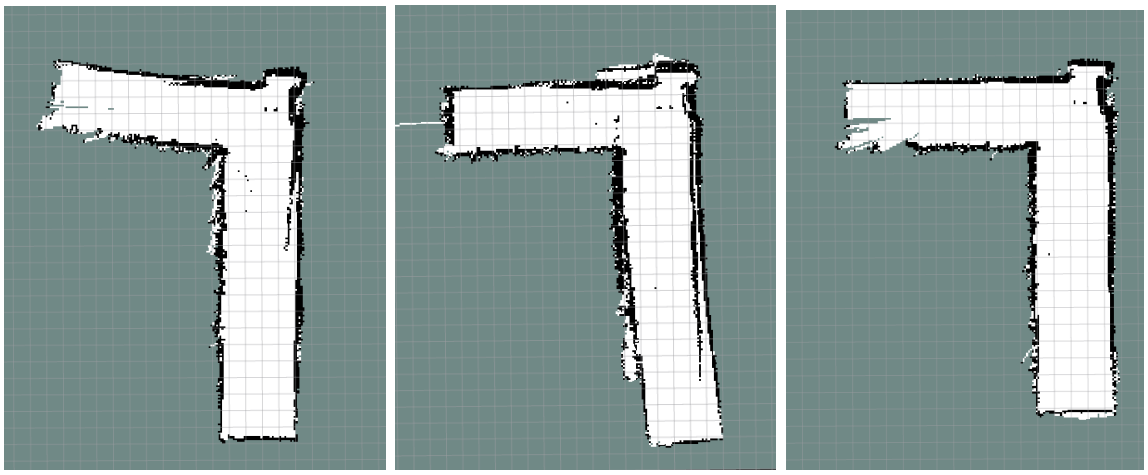
The following section includes a review of the results for the real-world hallway and lab tests as detailed in tables 4.4 and 4.5.

5.2.1 Hallway Tests Results

The hallway tests were conducted in the environment shown in Figure 4.5 and were conducted for robotic team sizes of $n = 2, 3, 4$. The tests were only benchmarked with the no coordination and shared map methods for a robotic team size of $n = 2$. The results for the experiments are found in Table 5.10. Some of the results in this section have been published in previous work by Goodwin and Nokleby [37]. From the results it can be seen that the k-means method greatly outperforms the no coordination and shared map method in time and average distance measures for Experiment 1. The k-means method outperforms both other methods reducing the completion time a

minimum of 44.9% and reducing the distance travelled by a minimum of 65.5%.

Figure 5.34 shows the merged map topics for the first hallway experiment. This includes the no coordination, shared map, and k-means methods for a robotic team size of $n = 2$. From the merged map topics it can be seen that the k-means map provided

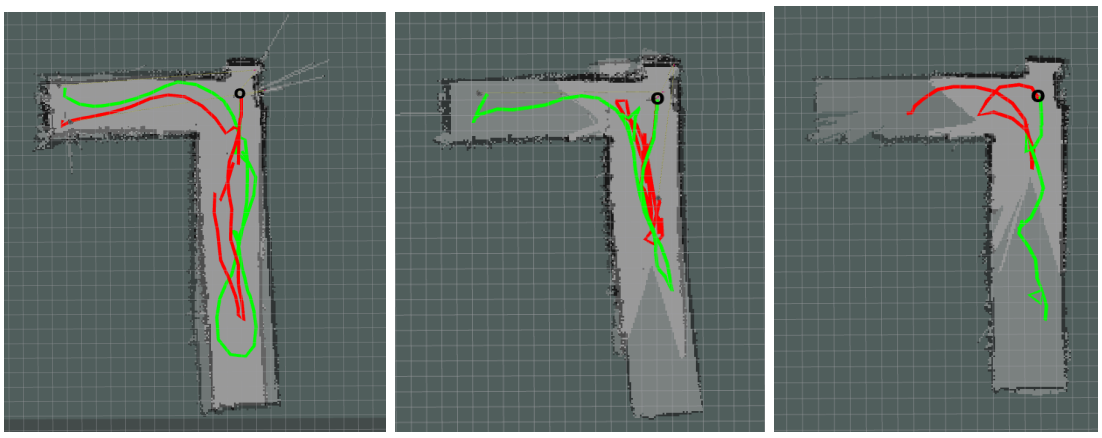


(a) No Coordination (b) Shared Map (c) K-means

Figure 5.34: Hallway Experiment 1: Merged Maps [37]

a more accurate map of the environment as the hallway forms a right angle.

The paths taken by the three methods are also presented for Experiment 1 in Figure 5.35. The starting points for the agents are labelled as a single circle on the map due



(a) No Coordination (b) Shared Map (c) K-means

Figure 5.35: Hallway Experiment 1: Path Comparisons [37]

to the agents beginning very close to each other. As can be seen in the figure, the

Table 5.11: Lab Experiment 1 Results

Method	**Time (s)**	**Avg. Dist. (m)**
no coordination	119.973	20.529
shared map	151.578	16.346
k-means	90.344	11.283

k-means method logically separates the hallway into two different sections and sends the two robots down the different sides of the hallway, as opposed to the two other methods. Additionally, the paths taken by the k-means method are much shorter by inspection compared to the no coordination and the shared map methods.

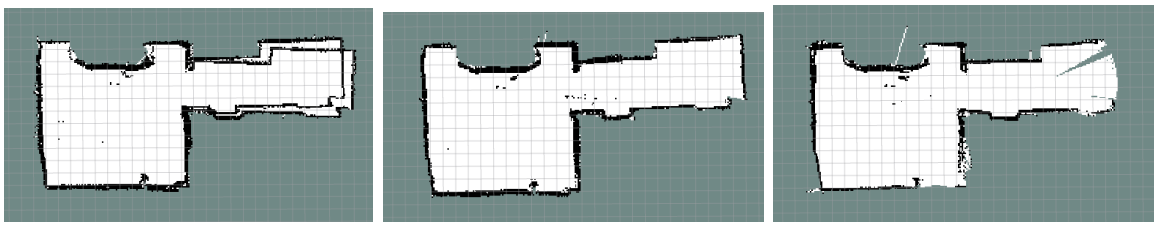
5.2.2 Lab Tests Results

The lab environment is shown in Figure 4.6 and was used to carry out three different experiments. The experiments were carried out for three robotic team sizes ($n = 2, 3, 4$) and all three methods were performed for each of the experiments. Recorded in the tables for each method is the time and average distance travelled by the agents for one run.

5.2.2.1 Lab Experiment 1 Results

The first experiment was for a robotic team size of $n = 2$ in the lab environment. In Table 5.11 the results for each of the different methods are recorded. From the table it can be seen that the k-means method reduces the time for the exploration task by a minimum of 25.7% and reduces the average distance travelled by a minimum of 31.0%. Another interesting note is that the shared map method takes more time than the no coordination method but the average distance travelled by each agent is smaller.

In Figure 5.36 the merged map topics for the Experiment 1 are presented. From the

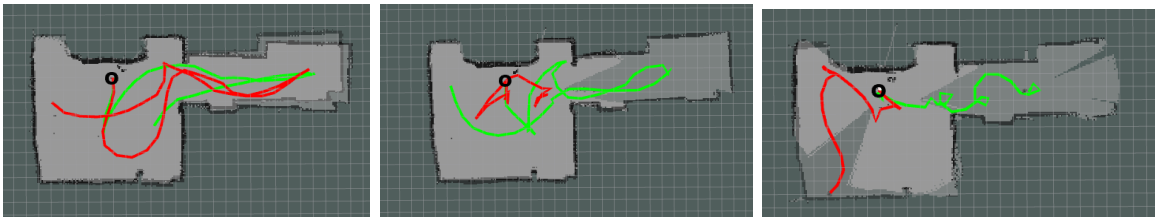


(a) No Coordination (b) Shared Map (c) K-means

Figure 5.36: Lab Experiment 1: Merged Maps

merged map topics it can be seen that the no coordination method provides the least accurate merged map while the shared map and k-means methods seem to perform similarly by inspection.

The paths for Experiment 1 are shown in Figure 5.37. The approximate starting



(a) No Coordination (b) Shared Map (c) K-means

Figure 5.37: Lab Experiment 1: Path Comparisons

points for the agents are shown as a single circle on the map due to the starting points being close together for these experiments. From the figures it can be seen that the k-means method provides the most segmentation of the map by sending one robot to the left of the environment and one to the right. The shared map method provides some segmentation as compared to the no coordination method. The figure shows that the shared map methods inadequate task allocation causes a lot of back and forth in the agents' paths.

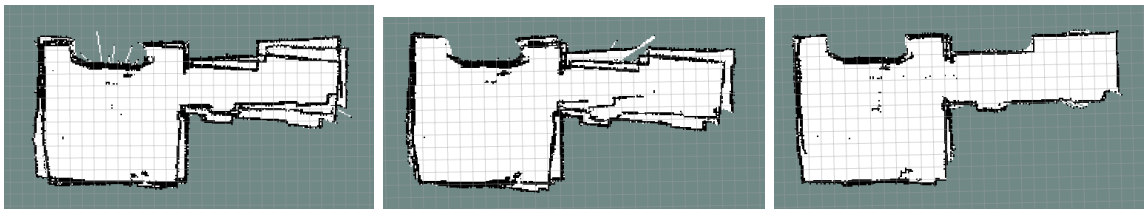
Table 5.12: Lab Experiment 2 Results

Method	**Time (s)**	**Avg. Dist. (m)**
no coordination	123.304	19.676
shared map	145.612	19.527
k-means	86.345	8.790

5.2.2.2 Lab Experiment 2 Results

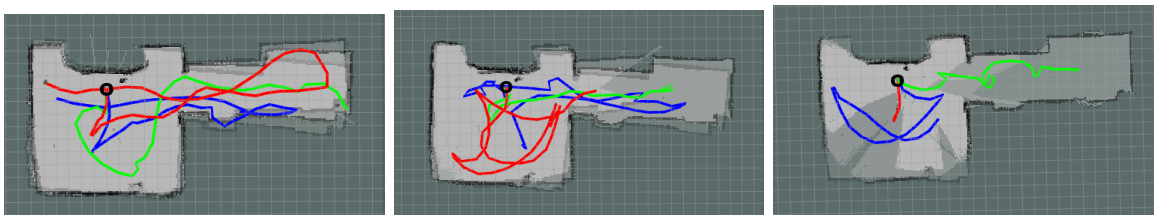
The second lab experiment used a robotic team size of $n = 3$. The time and distance results are recorded in table 5.12. From the table it can be seen that the k-means method outperforms both methods in time and distance measures. The k-means method reduced the completion time by a minimum of 30.0% and reduced the distance travelled by a minimum of 55.0%. The same trend is seen here as seen previously in Experiment 1 where the shared map method out performs the no coordination method in average distance, but not in time.

The merged maps are presented in Figure 5.38. This time the k-means method shows



(a) No Coordination (b) Shared Map (c) K-means

Figure 5.38: Lab Experiment 2: Merged Maps



(a) No Coordination (b) Shared Map (c) K-means

Figure 5.39: Lab Experiment 2: Path Comparisons

a significant improvement in the quality of the merged map over the no coordination and shared map methods.

Table 5.13: Lab Experiment 3 Results

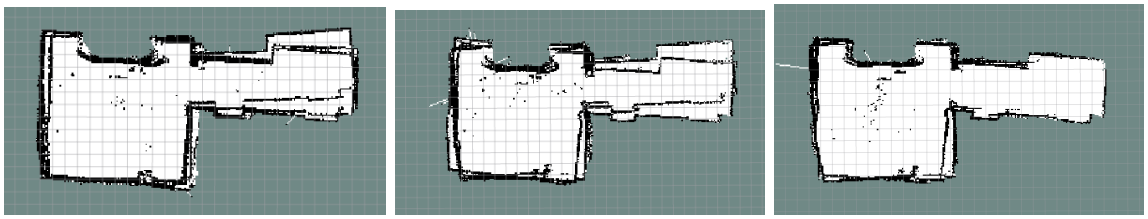
Method	**Time (s)**	**Avg. Dist. (m)**
no coordination	147.517	24.892
shared map	187.951	18.699
k-means	83.909	9.839

The paths for Experiment 2 taken by each agent are shown in Figure 5.39. From this figure it is easy to see that the k-means method provides a segmented exploration and greatly reduces the overlap seen by the shared map method.

5.2.2.3 Lab Experiment 3 Results

For the last of the lab experiments a robotic team size of $n = 4$ was used. The results for Experiment 3 are shown in Table 5.13. As seen in the previous experiments the k-means method outperforms the no coordination and shared map method. This time the k-means method reduced the completion time by a minimum of 43% and reduced the distance travelled by a minimum of 47.4%. The same trend as the other two lab experiments is observed in Experiment 3, the shared map method reduces the average distance travelled but not the completion time as compared to the no coordination method.

The merged maps for Experiment 3 can be seen in Figure 5.40. Looking at the merged

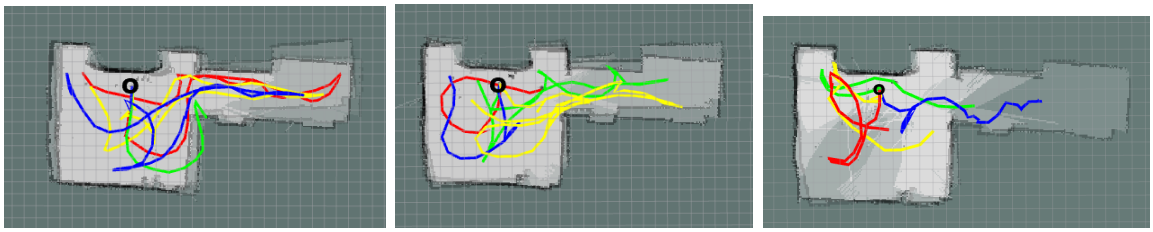


(a) No Coordination (b) Shared Map (c) K-means

Figure 5.40: Lab Experiment 3: Merged Maps

maps one can see that the k-means method provides a much cleaner map than the other methods.

The paths taken by each of the methods for Experiment 3 are seen in Figure 5.41. Again the k-means method shows a significant improvement over the shared map



(a) No Coordination (b) Shared Map (c) K-means

Figure 5.41: Lab Experiment 3: Path Comparisons

method. However, this time the segmentation of the map is not as clean as in the other experiments. This may be due to the geometry of the environment and the starting positions used for the experiment.

5.2.3 Real-World Tests Discussion

In this section a discussion of the different results for the real-world tests in the hallway and lab environments are detailed.

5.2.3.1 Hallway Tests Discussion

For the hallway tests a review of the performance of the k-means method for the different size robotic teams considered. The time and average distance is plotted for the experiments. In Figure 5.42 a graph is shown comparing the time for exploration versus the number of robots. From the figure it becomes clear for the hallway tests that the k-means method performed less efficiently in time measures as the robotic team size became larger.

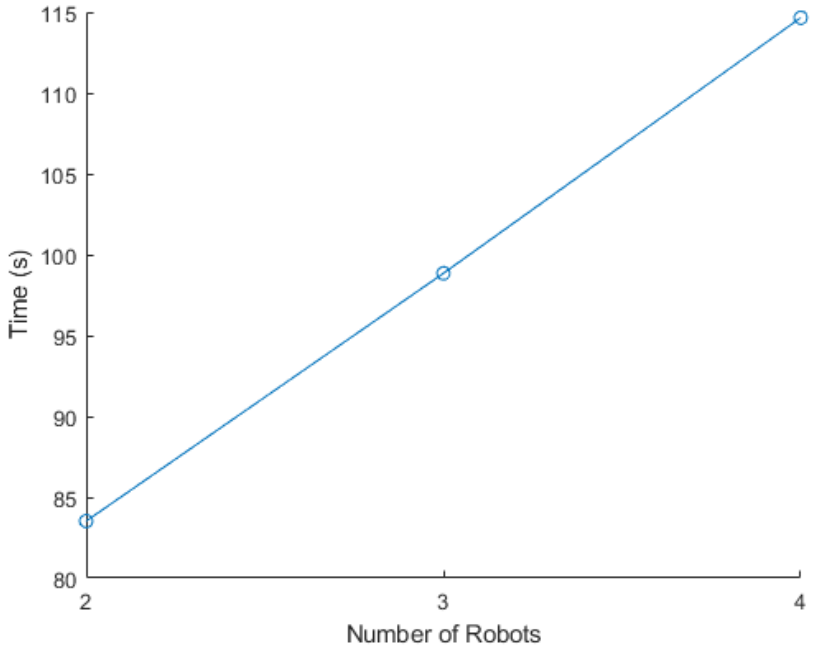


Figure 5.42: Hallways Experiments: Time

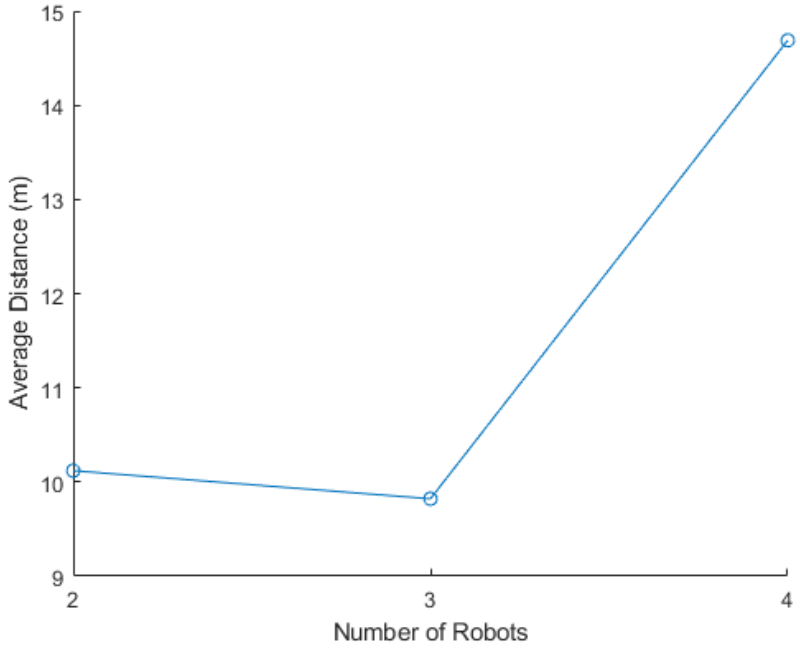
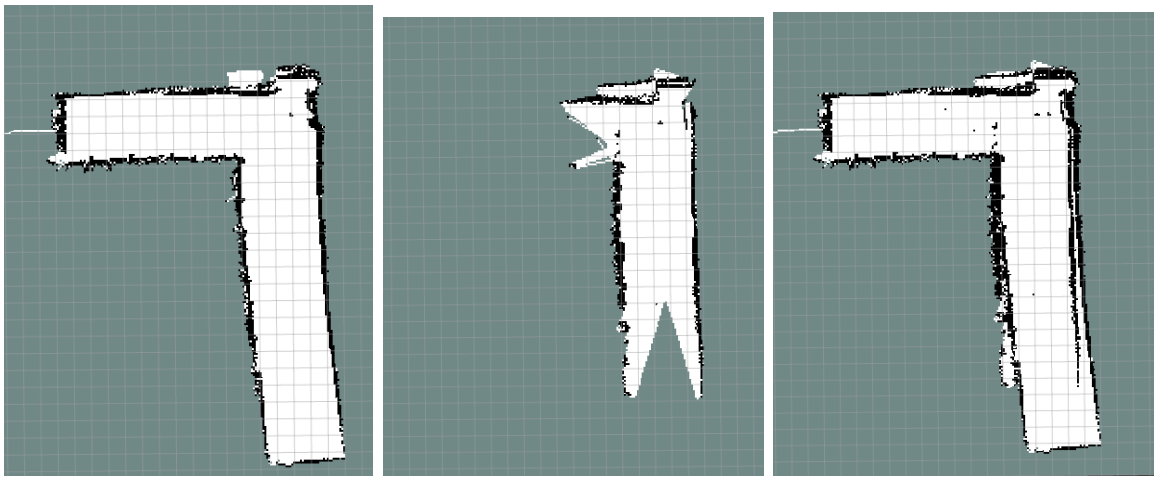


Figure 5.43: Hallway Experiments: Average Distance

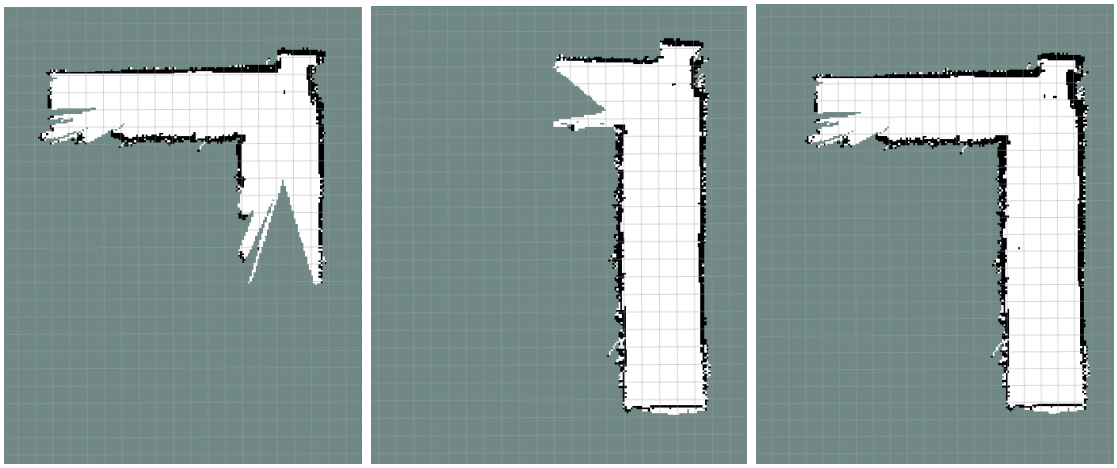
In Figure 5.43 the average distance travelled by each of the individual agents is plotted for the hallway experiments. In this figure the hallway experiment follows the

trend found in other experiments up to this point until the robotic team size became $n = 4$. A possible reason for this may be the fact that the hallway environment is too small to work efficiently for a robotic team size of greater than $n = 2$.

The hallway experiments were also benchmarked for a robotic team size of $n = 2$. The individual robot maps for the shared map and k-means methods are shown in Figure 5.44. As seen in the figure the shared map method provides a lot more overlap



(a) Shared Map Robot 1 Map (b) Shared Map Robot 2 Map (c) Shared Map Merged Map



(d) K-means Robot 1 Map (e) K-means Robot 2 Map (f) K-means Merged Map

Figure 5.44: Hallway Experiments: Map Coverage Comparisons [37]

in the map area that is covered, in fact Robot 1 completed the entire map by itself. The k-means method provides much better task allocation and less overlap in the map area covered creating a much quicker exploration process. The no coordination method is not included in this discussion because the no coordination method neces-

sitates that each agent completes its own individual map.

5.2.3.2 Lab Tests Discussion

For the lab experiments all three of the different methods were performed for three different sizes of robotic teams. The performance of the three methods is compared in this section by plotting the time and average distance comparisons for the methods. In Figure 5.45 the times for the different methods are compared over the different size robotic teams. The no coordination and shared map methods both trend upwards as

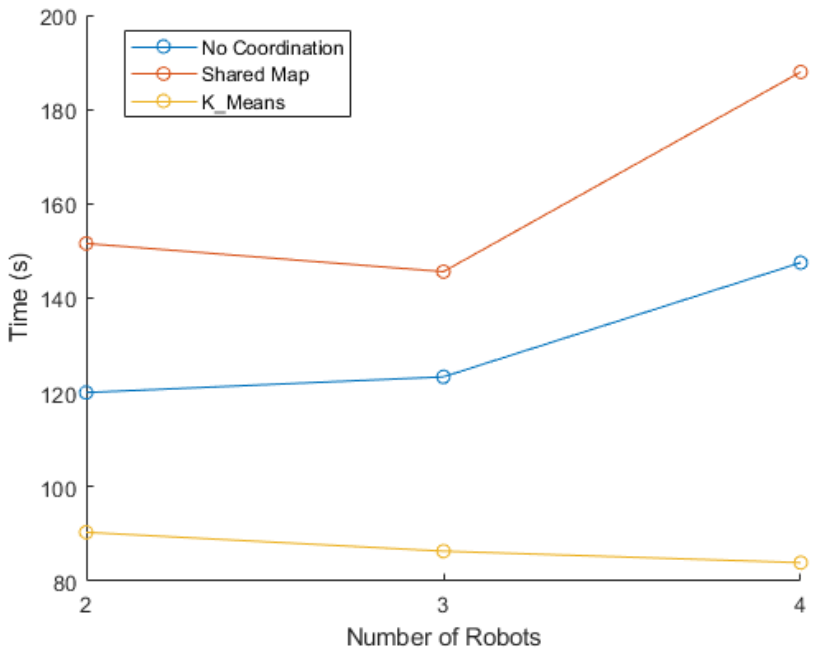


Figure 5.45: Lab Experiments: Time Comparisons

the size of the robotic team goes up. This is the same as the trends seen in all of the previous simulations and real-world experiments. However, in the lab experiments the k-means method decreases in time as the size of the robotic team increases.
The average distance for each of the agents is also plotted in Figure 5.46. The average distance for the no coordination method and the shared map method has a slightly upward trend as the size of robotic team gets larger. The k-means method also trends

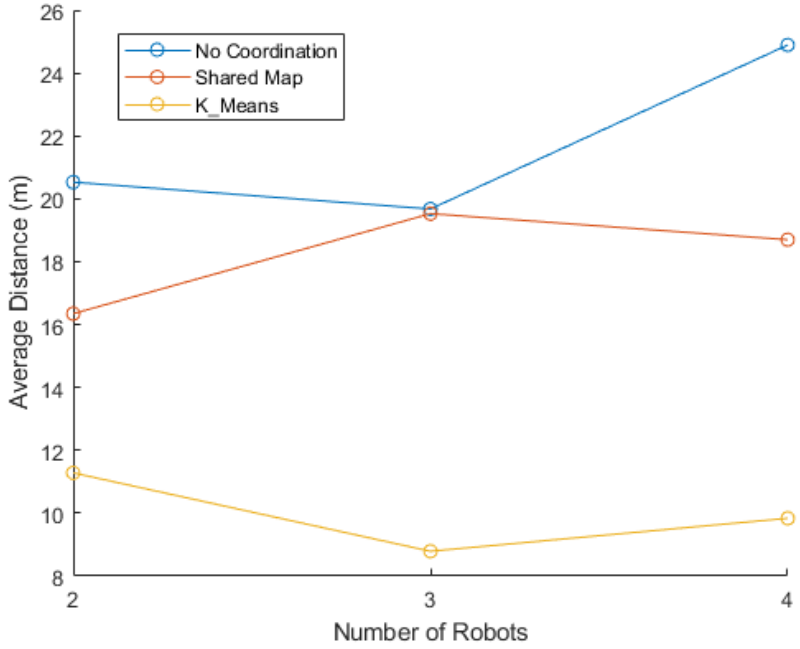


Figure 5.46: Lab Experiments: Average Distance Comparisons

slightly downward as the size of the robotic team increases.

5.3 Discussion

In summary, the developed k-means method was tested for simulated and real-world environments.

The benchmarking experiments were performed over three different environments, four different robotic team sizes, and with varied starting points. The results for the benchmarking tests showed the k-means method outperformed the other methods by reduced the exploration time by over 40% and up to 77% for all of the six experiments. The k-means method resulted in an average of 53% reduction in exploration time compared to the no coordination method and 55% reduction in exploration time compared to the shared map method. The k-means method also greatly outperformed both methods in reducing the average distance travelled by over 25% and up to 67%

for all six experiments. The k-means method reduced the distance travelled on average by 61% compared to the no coordination method and 50% compared to the shared map method. The k-means method also provided much lower standard deviation values in both time and distance measures proving the method is much more reliable and repeatable as compared to the no coordination and shared map methods. The k-means method also resulted in much less overlap in map area coverage as presented in the text.

For scalability Experiment 1 the results showed that starting the robots close together proves to be a much more difficult exploration task than starting the robots farther apart in both time and distance measures. However, the quality of merged maps increase when the robots are placed closer together to start. The time measures for far starting points seem to grow exponentially as the robotic team size increases. For close starting points the time increase is less predictable from the results seen, but appear to have a positive correlation with the size of the robotic team. The average distance for both far and close starting points decreases as the size of the robotic team increases. The standard deviation of the time for far starting points remains constant at a low value as the robotic team size increases, meaning the results are very repeatable. The standard deviation of the time for close starting points grows exponentially as the robotic team size increases meaning the results will vary greatly as the size in the robotic team increases. The standard deviation of the distance for the far starting points is lower than for close starting points, but they both seem to remain more constant throughout the increase in robotic team size.

For scalability Experiment 2 the results showed that the method is scalable to much larger robotic team sizes. However, the time needed for the exploration increases greatly before a robotic team size of $n = 5$ and then remains fairly constant. The average distance linearly decreases as the robotic team size increases.

The real-world results provide validation to a lot of the simulated experiment observations. The hallway tests showed that the average time for the exploration increases linearly as the robotic team size increases. The distance travelled does not decrease linearly as expected. This may be due to the size of the environment being too small

for the robotic team sizes being utilized. The k-means method also outperformed the other two methods in the hallway tests.

For the laboratory experiments all of the methods were tested for robotic team sizes of $n = 2, 3, 4$. The k-means method performed better in all of the experiments than the no coordination and shared map methods. The k-means method reduced the exploration time by an average reduction of 33% compared to the no coordination method and 45% compared to the shared map method. The k-means method reduced the distance travelled by an average reduction of 54% compared to the no coordination method and 44% compared to the shared map method. The paths taken for the k-means approach showed great improvement as compared to the shared map and no coordination methods. An important note is the time for the k-means method exploration decreased as the size of the robotic team increased, which is the first time this trend was observed.

Overall for all of the experiments presented the k-means method showed significant improvement as compared to the no coordination and shared map methods. The average time was reduced significantly in both simulated and real-world experiments, as well as the average distance travelled. The k-means method is proven to be scalable through the testing of robotic team sizes of $n = 2$ to $n = 9$. The effects of starting points as either close or far were studied and it was determined that the closer starting points creates a more difficult exploration in terms of time. However, the merged map topics are more accurate using close starting points as compared to far starting points. The k-means method presents a reliable system for segmentation of a previously unknown map, which decreases overlap in map coverage and in turn decreases the time and distances travelled required for each exploration task.

Chapter 6

Conclusions and Recommendations for Future Work

6.1 Conclusions

This thesis was created for the IDEaS project developed with the DND to research the development of effective human-machine cooperation with autonomous systems. A review of the literature was conducted and it was determined that frontier exploration is an efficient and time effective method for exploring new environments. However, the coordination of MRS exploration proved to be a more difficult task. In the literature a common method for task assignment in MRS is a cost-based analysis where each robot would be rewarded a goal point based upon presenting the lowest cost to reach a goal. The cost can be represented in different ways in the literature. Task assignment is handled either via a centralized, distributed or hybrid form of communication amongst the agents in the MRS. It was discovered that using a purely cost-based analysis for task allocation does not always provide an efficient exploration process. It was determined that segmentation of the map can provide a good alternative to purely cost-based task allocation scheme. Segmenting a map

allows for less overlap in map coverage by the robots in a MRS which leads to less elapsed time for the exploration task and a smaller distances travelled by each robot. Providing a method to segment the map without previously knowing the map beforehand is a difficult task. A k-means method to segment the map is determined to be a fast and simple approach to segmenting an unknown environment.

A framework was developed using ROS and open source TB3 Burgers. A series of already developed packages in ROS were used for SLAM, autonomous navigation and map merging. Explore_lite was modified to be used in MRS where each agent will broadcast points of interest to a topic. A new method called the "assigner" method was developed to receive these points and redistribute them to the different robots in the system. The assigner method consists of multiple modules to facilitate this task. The first is the filtering module. The filtering module filters points of interest against the already known global map using a neighbourhood sampling method. The second is the k-means method. The k-means method takes in all unknown points of interest and clusters them into n clusters, where n is the number of agents. Thirdly an assignment method is used to assign the clusters using a simple cost based analysis.

Experiments were designed for the testing of this method in both simulated and real-world environments. For simulations three different environments were used to show the versatility of the method and two different types of tests were created. The first tests created were bench-marking tests. The bench-marking tests were developed to show the improvement of the k-means method over the no coordination and shared map methods. The no coordination method seeks to provide a baseline value for how long it would take an individual agent to complete the entire exploration task. The shared map method seeks to show the effects of sharing a map without adequate task allocation for MRS exploration. Testing was designed to be performed over six different experiments with both close and far starting points for variability. The scalability tests were designed to showcase the scalability of the system and the effect of starting location on the performance. Two different environments were used with different sizes of robotic teams ranging from $n = 2$ to $n = 9$ and varied starting points. Real-world tests were carried out first with preliminary testing in a hallway environment

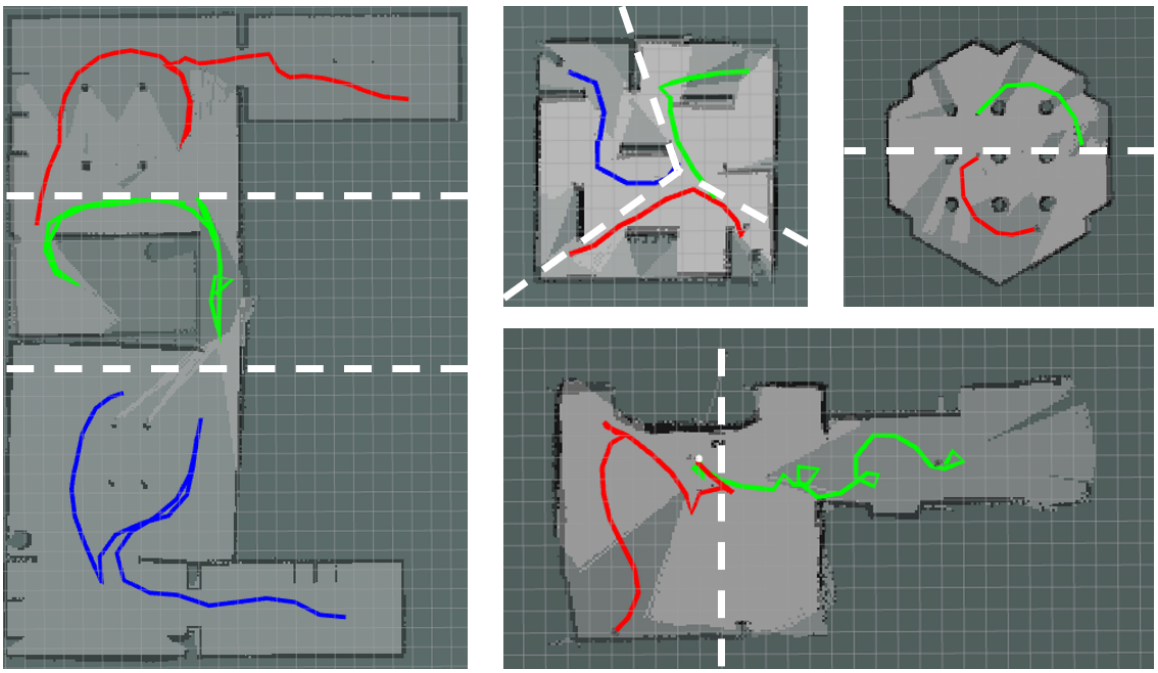


Figure 6.1: Examples of K-means Segmentation of the Map

and secondary testing in a lab environment. Both tests featured different sizes of robotic teams ($n = 2, 3, 4$). Most of the real-world experiments were bench-marked against the no coordination and shared map methods to validate the findings.

The results for the different simulated tests showed that the proposed k-means method provides good segmentation of a previously unknown map. In Figure 6.1 some examples from the different experiments are shown. In the figure white dotted lines are used to show how the map is being segmented for the different agents in the system. When compared to the other methods in the bench-marking experiments, the k-means method greatly outperformed both the no coordination and shared map methods. The k-means method reduced the time needed for the simulated exploration tasks by over 40% for all experiments and up to 70%. The method reduced the average distance travelled by each robot by over 25% for all experiments and up to 67%. The method also produced a much smaller standard deviation in all of the time and distance values, proving the method to be much more repeatable and reliable. The scalability tests proved that the method can be scaled to many different sizes of robotic teams with tests featuring up to $n = 9$ robots. The time, however, in-

creased as the robotic team size got larger. The distance travelled linearly decreased as the robotic team size increased. It was also observed that closer starting points generated more accurately merged maps, but farther starting points resulted in faster exploration times.

The results for the real-world tests proved the reliability of the proposed method. The results for the hallway experiments showed improvement over the other methods for exploration. The k-means method did not perform as well when the robotic team size increased, as the time and distance travelled increased when the robotic team became $n = 4$. This was theorized to be due to the size of the environment being too small.

The laboratory experiments proved that the k-means method improved the time results by a minimum of 25% for all experiments and up to 55%. The time taken for the k-means method exploration task decreased as the robotic team size increased for the first time in the results. This is a drastic improvement as compared to the other two methods that increase in time as the robotic team size increases. The average distance measure for the k-means method decreased as the size of the robotic team increased, while both the no coordination method and shared map method increase.

The k-means method proves to be a reliable, scalable, and time-efficient exploration method for MRS. The method performed better in time and distance measures than the no coordination and shared map methods in every experiment performed.

6.2 Future Work and Recommendations

The developed k-means method has been proven to provide an efficient method for segmentation of a previously unknown map in real-time. The method faces some challenges to providing an even better exploration task. A few recommendations for improving the method are outlined below.

The method requires a more efficient cluster assignment method. The current method used for the purposes of this thesis has each cluster assigned iteratively to the closest agent, however, there are cases when this leads to agents not being assigned to the most cost efficient cluster from a global perspective. Methods such as the Hungarian method [31] for task assignment could be implemented to determine better task assignments. Though this method requires longer computational times, which could greatly increase the run time of the method in real-time applications.

The next improvement would be to incorporate a quantifiable level of trust displayed by each of the agents. At times agents could encounter errors in their odometry which can pose problems when merging the maps into one global representation. If a level of trust was defined, the central agent could determine if an agent is not displaying an accurate map representation. On a similar note, a level of trust in each of the agents could help define if one of the agents lost connection or failed to start. At times when running simulations different agents would not start when the exploration was commenced or would get caught on different objects during the exploration task. A more robust system could determine whether the agents are online and assign the tasks appropriately while some of the agents are offline.

Another suggestion for future work is to incorporate the use of Unmanned Aerial Vehicles (UAVs) which provide important obstacle information for ground vehicles while online. UAVs could also be used to help fix dead-reckoning errors for the individual agents which would improve the accuracy of the merged map topic.

To improve the global map, a graphical map merging technique could be utilized after finishing the exploration process for a more accurate final merged map.

References

[1] J. Forlizzi and C. DiSalvo, "Service robots in the domestic environment: a study of the roomba vacuum in the home," in *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-robot interaction*, 2006, pp. 258–265.

[2] V. Digani, L. Sabattini, C. Secchi, and C. Fantuzzi, "Ensemble coordination approach in multi-agv systems applied to industrial warehouses," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 922–934, 2015.

[3] M. Cardona, F. Cortez, A. Palacios, and K. Cerros, "Mobile robots application against covid-19 pandemic," in *2020 IEEE ANDESCON*, pp. 1–5.

[4] C. Hu, C. Hu, D. He, and Q. Gu, "A new ros-based hybrid architecture for heterogeneous multi-robot systems," in *The 27th Chinese Control and Decision Conference*. IEEE, 2015, pp. 4721–4726.

[5] A. El Shenawy, K. Mohamed, and H. M. Harb, "Exploration strategies of coordinated multi-robot system: A comparative study," in *IAES International Journal of Robotics and Automation (IJRA)*, vol. 7, no. 1, 2018, pp. 48–58.

[6] B. Yamauchi, "Frontier-based exploration using multiple robots," in *Proceedings of the Second International Conference on Autonomous Agents*, 1998, pp. 47–53.

[7] Y. Wang, A. Liang, and H. Guan, "Frontier-based multi-robot map exploration using particle swarm optimization," in *2011 IEEE Symposium on Swarm Intelligence*, pp. 1–6.

[8] J. De Hoog, S. Cameron, and A. Visser, "Role-based autonomous multi-robot exploration," in *2009 Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns*, pp. 482–487.

[9] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*, pp. 146–151.

[10] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes, "Coordination for multi-robot exploration and mapping," in *Proceedings of the AAAI National Conference on Artificial Intelligence*, 2000, pp. 852–858.

[11] A. Bautin, O. Simonin, and F. Charpillet, "Minpos: A novel frontier allocation algorithm for multi-robot exploration," in *International Conference on Intelligent Robotics and Applications*. Springer, 2012, pp. 496–508.

[12] R. Zlot, A. Stentz, M. B. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," in *2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 3, pp. 3016–3023.

[13] R. S. D. Muddu, D. Wu, and L. Wu, "A frontier based multi-robot approach for coverage of unknown environments," in *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2015, pp. 72–77.

[14] M. N. Rooker and A. Birk, "Multi-robot exploration under the constraints of wireless networking," *Control Engineering Practice*, vol. 15, no. 4, pp. 435–445, 2007.

[15] M. Corah and N. Michael, "Efficient online multi-robot exploration via distributed sequential greedy assignment." in *Robotics: Science and Systems*, vol. 13, 2017.

[16] N. Mahdoui, V. Frémont, and E. Natalizio, "Cooperative frontier-based exploration strategy for multi-robot system," in *2018 13th Annual Conference on System of Systems Engineering (SoSE)*. IEEE, 2018, pp. 203–210.

[17] J. P. Desai, J. P. Ostrowski, and V. Kumar, "Modeling and control of formations of nonholonomic mobile robots," *IEEE transactions on Robotics and Automation*, vol. 17, no. 6, pp. 905–908, 2001.

[18] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 1987, pp. 25–34.

[19] N. Bouraqadi, A. Doniec, and E. de Douai, "Flocking-based multi-robot exploration," in *National Conference on Control Architectures of Robots*. Citeseer, 2009, pp. 1–8.

[20] A. Kumar, S. Sharma, R. Tiwari, and S. Majumdar, "Area exploration by flocking of multi robot," *Procedia Engineering*, vol. 41, pp. 377–382, 2012.

[21] G. Li, D. Zhang, and Y. Shi, "An unknown environment exploration strategy for swarm robotics based on brain storm optimization algorithm," in *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2019, pp. 1044–1051.

[22] M. S. Couceiro, R. P. Rocha, and N. M. Ferreira, "A novel multi-robot exploration approach based on particle swarm optimization algorithms," in *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*. IEEE, 2011, pp. 327–332.

[23] A. S. Kumar, G. Manikutty, R. R. Bhavani, and M. S. Couceiro, "Search and rescue operations using robotic darwinian particle swarm optimization," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2017, pp. 1839–1843.

[24] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 376–386, 2005.

[25] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart, "Distributed multirobot exploration and mapping," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1325–1339, 2006.

[26] A. Pal, R. Tiwari, and A. Shukla, "Multi-robot exploration in wireless environments," *Cognitive Computation*, vol. 4, no. 4, pp. 526–542, 2012.

[27] M. N. Rooker and A. Birk, "Multi-robot exploration under the constraints of wireless networking," *Control Engineering Practice*, vol. 15, no. 4, pp. 435–445, 2007.

[28] M. N. Rooker and A. Birk, "Communicative exploration with robot packs," in *Robot Soccer World Cup*. Springer, 2005, pp. 267–278.

[29] B. P. Gerkey and M. J. Mataric, "Sold!: Auction methods for multirobot coordination," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 758–768, 2002.

[30] W. Sheng, Q. Yang, J. Tan, and N. Xi, "Distributed multi-robot coordination in area exploration," *Robotics and Autonomous Systems*, vol. 54, no. 12, pp. 945–955, 2006.

[31] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[32] K. M. Wurm, C. Stachniss, and W. Burgard, "Coordinated multi-robot exploration using a segmentation of the environment," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 1160–1165.

[33] M. Elango, S. Nachiappan, and M. K. Tiwari, "Balancing task allocation in multi-robot systems using k-means clustering and auction based mechanisms," *Expert Systems with Applications*, vol. 38, no. 6, pp. 6486–6491, 2011.

[34] A. Solanas and M. A. Garcia, "Coordinated multi-robot exploration through unsupervised clustering of unknown space," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 1. IEEE, 2004, pp. 717–721.

[35] D. Puig, M. A. García, and L. Wu, "A new global optimization strategy for coordinated multi-robot exploration: Development and comparative evaluation," *Robotics and Autonomous Systems*, vol. 59, no. 9, pp. 635–653, 2011.

[36] J. Faigl, M. Kulich, and L. Přeučil, "Goal assignment using distance cost in multi-robot exploration," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 3741–3746.

[37] L. Goodwin and S. Nokleby, "A k-means clustering approach to segmentation of maps for task allocation in multi-robot systems exploration of unknown environments," in *Proceedings of the 2022 USCToMM Mechanisms, Machines, and Mechatronics Symposium*, 2022.

[38] H. J. Chang, C. G. Lee, Y. C. Hu, and Y.-H. Lu, "Multi-robot slam with topological/metric maps," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1467–1472.

[39] D. Meier, C. Stachniss, and W. Burgard, "Coordinating multiple robots during exploration under communication with limited bandwidth," in *Proceedings of the European Conference on Mobile Robots (ECMR)*, 2005, pp. 26–31.

[40] J. De Hoog, S. Cameron, and A. Visser, "Selection of rendezvous points for multi-robot exploration in dynamic environments," in *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2010.

[41] K. Konolige, D. Fox, B. Limketkai, J. Ko, and B. Stewart, "Map merging for distributed robot navigation," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, vol. 1, pp. 212–217.

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

[43] L. Goodwin and S. Nokleby, "Multi-robot exploration of unknown environments," in *Proceedings of the 2021 CCToMM Mechanisms, Machines, and Mechatronics Symposium*, 2021.

[44] ROBOTIS, "ROBOTIS E-Manual: TurtleBot3 Burger." [Online]. Available: https://emanual.robotis.com/docs/en/platform/turtlebot3/specifications/#specifications

[45] ROBOTIS, "Turtlebot3." [Online]. Available: https://github.com/ROBOTIS-GIT/turtlebot3.git

[46] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, "The office marathon: Robust navigation in an indoor office environment," in *International Conference on Robotics and Automation*, 2010.

[47] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[48] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[49] B. Gerkey, "Gmapping." [Online]. Available: https://github.com/ros-perception/slam_gmapping.git

[50] J. Hörner, "Map-merging for multi-robot system," Bachelor's thesis, Charles University in Prague, Faculty of Mathematics and Physics, Prague, 2016. [Online]. Available: https://is.cuni.cz/webapps/zzp/detail/174125/

[51] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[52] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, pp. 2149–2154.

Appendix A

Experimental Results

The following section contains all the results for the simulated experiments. Some of these results were published in earlier work by Goodwin and Nokleby [37].

A.1 Simulation: Benchmarking Tests

In the next sections the experimental results can be see in full detail for all of the benchmarking experiments. In these tables distance is abbreviated as dist., the section experiment number corresponds to the table found in Chapter 4, Table 4.1. Some runs do not finish and have been marked as DNF.

A.1.1 Benchmarking Experiment 1

Table A.1: Benchmarking Experiment 1 No Coordination Results

Run	Time 1 (s)	Time 2 (s)	Dist. 1 (m)	Dist. 2 (m)
1	126.403	167.765	5.743	6.671
2	148.616	98.265	6.147	4.069
3	156.504	93.814	6.669	3.671
4	181.345	181.351	6.557	5.973
5	107.306	144.136	3.761	6.156

Table A.2: Benchmarking Experiment 1 Shared Map Results

Run	Time (s)	Dist. 1 (m)	Dist. 2 (m)
1	103.979	3.669	3.645
2	125.980	4.684	3.400
3	155.668	5.788	6.004
4	130.691	3.325	5.785
5	184.611	3.592	6.436

Table A.3: Benchmarking Experiment 1 K-Means Method Results

Run	Time (s)	Dist. 1 (m)	Dist. 2 (m)
1	84.411	3.195	2.742
2	78.682	3.257	2.423
3	63.687	3.863	4.037
4	83.215	2.582	2.542
5	74.803	2.501	2.457

A.1.2 Benchmarking Experiment 2

Table A.4: Benchmarking Experiment 2 No Coordination Results

Run	Time 1 (s)	Time 2 (s)	Dist. 1 (m)	Dist. 2 (m)
1	193.076	389.824	10.924	7.692
2	397.650	347.483	11.656	7.812
3	184.937	305.670	9.686	12.635
4	333.480	186.909	14.358	10.876
5	212.152	117.469	16.620	9.288

Table A.5: Benchmarking Experiment 2 Shared Map Results

Run	Time (s)	Dist. 1 (m)	Dist. 2 (m)
1	268.546	13.074	6.607
2	236.303	15.251	7.659
3	268.303	7.590	9.164
4	339.603	9.850	11.221
5	200.142	10.765	5.953

Table A.6: Benchmarking Experiment 2 K-Means Method Results

Run	Time (s)	Dist. 1 (m)	Dist. 2 (m)
1	197.523	5.321	6.142
2	150.809	5.151	5.563
3	151.410	3.809	4.301
4	127.398	3.254	3.442
5	112.043	4.175	4.518

A.1.3 Benchmarking Experiment 3

Table A.7: Benchmarking Experiment 3 No Coordination Results

Run	Time 1 (s)	Time 2 (s)	Time 3 (s)	Dist. 1 (m)	Dist. 2 (m)	Dist. 3 (m)
1	481.186	458.043	323.098	12.425	9.164	9.414
2	436.921	438.895	198.725	10.631	10.349	8.155
3	632.085	326.610	297.228	17.727	11.716	8.383
4	518.516	436.906	196.767	16.625	15.229	8.656
5	616.417	361.164	656.875	13.101	12.385	12.660

Table A.8: Benchmarking Experiment 3 Shared Map Results

Run	Time (s)	Dist. 1 (m)	Dist. 2 (m)	Dist. 3 (m)
1	574.128	9.125	13.326	12.885
2	736.834	11.147	7.955	10.499
3	275.338	8.103	7.597	7.400
4	506.251	13.630	0.266	10.923
5	585.945	20.084	10.700	12.015

Table A.9: Benchmarking Experiment 3 K-Means Method Results

Run	Time (s)	Dist. 1 (m)	Dist. 2 (m)	Dist. 3 (m)
1	144.000	4.342	3.740	4.400
2	198.749	5.484	5.293	5.180
3	152.319	4.470	1.028	1.834
4	196.887	2.456	2.859	4.136
5	180.432	5.086	3.011	3.348

A.1.4 Benchmarking Experiment 4

Table A.10: Benchmarking Experiment 4 No Coordination Results

Run	Time 1 (s)	Time 2 (s)	Time 3 (s)
1	191.306	356.085	404.429
2	326.141	339.334	350.284
3	663.799	700.625	217.866
4	325.779	460.704	376.063
5	373.672	501.238	501.211

Dist. 1 (m)	Dist. 2 (m)	Dist. 3 (m)
6.325	9.954	11.963
7.192	9.166	9.692
13.295	12.570	6.831
7.153	12.664	12.998
7.977	16.161	12.734

Table A.11: Benchmarking Experiment 4 Shared Map Results

Run	Time (s)	Dist. 1 (m)	Dist. 2 (m)	Dist. 3 (m)
1	255.291	7.587	8.212	7.773
2	544.438	15.831	11.310	9.234
3	296.368	7.625	9.128	7.686
4	430.286	6.515	11.739	14.922
5	173.158	5.371	5.855	5.681

Table A.12: Benchmarking Experiment 4 K-Means Method Results

Run	Time (s)	Dist. 1 (m)	Dist. 2 (m)	Dist. 3 (m)
1	97.109	3.467	3.542	2.829
2	150.085	3.667	3.487	3.287
3	177.377	4.041	4.371	4.245
4	201.223	3.883	4.077	2.949
5	176.525	4.354	4.317	3.817

A.1.5 Benchmarking Experiment 5

Table A.13: Benchmarking Experiment 5 No Coordination Results

Run	Time 1 (s)	Time 2 (s)	Time 3 (s)	Time 4 (s)
1	575.692	692.413	462.168	970.369
2	988.963	368.623	600.941	820.745
3	714.320	840.132	554.255	598.941
4	DNF	426.033	516.71	DNF
5	DNF	469.798	617.07	DNF

Dist. 1 (m)	Dist. 2 (m)	Dist. 3 (m)	Dist. 4 (m)
12.161	10.253	10.736	12.664
18.117	8.454	10.736	12.513
17.413	11.972	8.846	13.006
DNF	10.732	7.168	DNF
DNF	9.427	10.684	DNF

Table A.14: Benchmarking Experiment 5 Shared Map Results

Run	Time (s)	Dist. 1 (m)	Dist. 2 (m)	Dist. 3 (m)	Dist. 4 (m)
1	856.818	12.421	9.700	7.195	8.156
2	1160.073	19.161	8.009	6.158	7.750
3	1352.023	9.463	10.920	4.631	15.626
4	1492.512	6.383	12.572	9.600	16.498
5	1581.236	16.859	11.000	7.514	20.634

Table A.15: Benchmarking Experiment 5 K-Means Method Results

Run	Time (s)	Dist. 1 (m)	Dist. 2 (m)	Dist. 3 (m)	Dist. 4 (m)
1	217.826	3.611	3.031	3.143	3.397
2	323.279	6.335	1.832	1.331	2.311
3	322.528	6.456	5.721	5.292	3.916
4	339.075	6.260	5.781	5.231	4.578
5	276.768	4.761	3.733	3.757	4.039

A.1.6 Benchmarking Experiment 6

Table A.16: Benchmarking Experiment 6 No Coordination Results

Run	Time 1 (s)	Time 2 (s)	Time 3 (s)
1	1026.363	2214.398	1210.414
2	1111.643	1534.935	1589.646
3	1080.655	810.825	1341.455
4	1050.525	1956.607	2515.277
5	1210.405	2550.222	1528.958

Dist. 1 (m)	Dist. 2 (m)	Dist. 3 (m)
43.947	79.565	52.610
43.941	62.375	68.828
53.601	38.244	67.877
45.541	73.120	93.870
56.403	86.298	58.519

Table A.17: Benchmarking Experiment 6 Shared Map Results

Run	Time (s)	Dist. 1 (m)	Dist. 2 (m)	Dist. 3 (m)
1	DNF	21.612	DNF	16.036
2	1168.062	24.003	48.090	18.546
3	604.339	25.262	13.582	20.986
4	1599.038	25.929	65.692	17.883
5	1304.657	22.475	58.781	16.106

Table A.18: Benchmarking Experiment 6 K-Means Method Results

Run	Time (s)	Dist. 1 (m)	Dist. 2 (m)	Dist. 3 (m)
1	631.512	21.020	19.260	18.296
2	590.789	16.514	18.589	19.937
3	661.409	25.296	17.625	20.946
4	678.239	23.091	23.842	26.961
5	726.270	22.902	11.181	17.806

A.2 Simulation: Scalability Tests

In the next sections the experimental results can be seen in full detail for all of the scalability experiments. In these tables distance is abbreviated as dist., the test number and experiment number correspond to the tables found in Chapter 4, tables 4.2 and 4.3.

A.2.1 Scalability Experiment 1.1

Table A.19: Scalability Experiment 1.1a K-means Results Far Start

Run	**Time (s)**	**Dist. 1 (m)**	**Dist. 2 (m)**
1	197.523	5.321	6.142
2	150.809	5.151	5.563
3	151.410	3.809	4.301
4	127.398	3.254	3.442
5	112.043	4.175	4.518

Table A.20: Scalability Experiment 1.1b K-means Results Close Start

Run	**Time (s)**	**Dist. 1 (m)**	**Dist. 2 (m)**
1	180.500	5.526	6.568
2	235.172	7.361	10.047
3	223.119	7.983	7.403
4	229.276	7.185	8.096
5	216.920	5.668	9.390

A.2.2 Scalability Experiment 1.2

Table A.21: Scalability Experiment 1.2a K-means Results Far Start

Run	Time (s)	Dist. 1 (m)	Dist. 2 (m)	Dist. 3 (m)
1	97.109	3.467	3.542	2.829
2	150.085	3.667	3.487	3.287
3	177.377	4.041	4.371	2.949
4	201.223	3.883	4.077	2.949
5	176.525	4.354	4.317	3.817

Table A.22: Scalability Experiment 1.2b K-means Results Close Start

Run	Time (s)	Dist. 1 (m)	Dist. 2 (m)	Dist. 3 (m)
1	144.000	4.342	3.740	4.400
2	198.749	5.484	5.293	5.180
3	152.319	4.470	1.028	1.834
4	196.887	2.456	2.859	4.136
5	180.432	5.086	3.011	3.348

A.2.3 Scalability Experiment 1.3

Table A.23: Scalability Experiment 1.3a K-means Results Far Start

Run	Time (s)	Dist. 1 (m)	Dist. 2 (m)	Dist. 3 (m)	Dist. 4 (m)
1	178.576	3.209	2.940	0.072	0.651
2	162.647	3.200	2.866	1.144	0.641
3	261.701	2.502	5.040	3.203	2.979
4	137.542	1.150	1.577	1.067	0.000
5	236.738	4.164	1.939	3.874	3.018

Table A.24: Scalability Experiment 1.3b K-means Results Close Start

Run	Time (s)	Dist. 1 (m)	Dist. 2 (m)	Dist. 3 (m)	Dist. 4 (m)
1	217.826	3.611	3.031	3.143	3.397
2	323.279	6.335	1.832	1.331	2.311
3	322.528	6.456	5.721	5.292	3.916
4	339.075	6.260	5.781	5.231	4.578
5	276.768	4.761	3.733	3.757	4.039

A.2.4 Scalability Experiment 1.4

Table A.25: Scalability Experiment 1.4a K-means Results Far Start

Run	Time (s)	Dist. 1 (m)	Dist. 2 (m)	Dist. 3 (m)	Dist. 4 (m)	Dist. 5 (m)
1	260.000	4.193	3.637	1.355	2.240	2.761
2	283.483	1.845	3.250	1.383	1.936	2.412
3	376.861	1.517	4.537	1.832	2.250	2.943
4	321.769	1.647	4.622	1.508	2.782	2.985
5	276.773	2.565	2.054	0.000	0.706	1.692

Table A.26: Scalability Experiment 1.4b K-means Results Close Start

Run	Time (s)	Dist. 1 (m)	Dist. 2 (m)	Dist. 3 (m)	Dist. 4 (m)	Dist. 5 (m)
1	329.303	4.626	3.073	3.941	1.603	3.086
2	349.650	4.405	4.521	4.688	0.504	2.953
3	549.251	3.982	5.840	5.566	1.644	5.144
4	278.461	5.034	3.763	4.425	4.217	2.521
5	446.723	4.127	6.698	3.363	5.904	6.186

A.2.5 Scalability Experiment 2.1

Table A.27: Scalability Experiment 2.1 K-means Results

Run	Time (s)	Dist. 1 (m)	Dist. 2 (m)	Dist. 3 (m)
1	631.512	21.020	19.260	18.296
2	590.789	16.514	18.589	19.937
3	661.409	25.296	17.625	20.946
4	678.239	23.091	23.842	26.961
5	726.270	22.902	11.181	17.806

A.2.6 Scalability Experiment 2.2

Table A.28: Scalability Experiment 2.2 K-means Results

Run	Time (s)	Dist. 1 (m)	Dist. 2 (m)	Dist. 3 (m)	Dist. 4 (m)	Dist. 5 (m)
1	498.990	9.541	7.223	9.472	8.463	7.038
2	955.145	17.910	15.414	16.236	14.358	15.122
3	769.694	15.782	8.896	11.905	14.625	14.525
4	778.879	15.032	13.905	13.383	14.805	15.523
5	686.317	13.078	13.030	13.636	15.706	13.358

A.2.7 Scalability Experiment 2.3

Table A.29: Scalability Experiment 2.3 K-means Results

Run	Time (s)	Dist. 1 (m)	Dist. 2 (m)	Dist. 3 (m)	Dist. 4 (m)	Dist. 5 (m)	Dist. 6 (m)	Dist. 7 (m)
1	986.417	10.088	7.863	14.642	12.268	6.609	13.939	12.692
2	669.178	9.362	8.676	7.233	8.604	8.125	7.200	9.127
3	533.133	7.349	6.949	8.623	8.188	8.007	7.230	9.347
4	585.089	8.121	6.771	7.757	7.580	4.822	3.854	7.903
5	950.463	8.527	10.903	12.461	11.481	12.287	8.313	13.328

A.2.8 Scalability Experiment 2.4

Table A.30: Scalability Experiment 2.4 K-means Results

Run	Time (s)	Dist. 1 (m)	Dist. 2 (m)	Dist. 3 (m)	Dist. 4 (m)	Dist. 5 (m)	Dist. 6 (m)	Dist. 7 (m)	Dist. 8 (m)	Dist. 9 (m)
1	841.843	3.864	4.260	6.780	1.682	5.867	4.852	6.143	3.230	5.880
2	839.619	8.658	7.397	8.893	7.586	5.547	4.631	5.714	5.844	5.262
3	540.166	3.411	4.238	4.763	3.984	3.301	1.962	3.147	2.877	3.244
4	952.505	7.746	8.113	9.146	6.941	6.316	7.530	7.423	7.532	9.209
5	543.364	4.406	3.767	4.201	3.555	1.980	3.531	3.874	3.406	3.862