Design and Implementation of Efficient Coordination Algorithms for Multi-Robot Systems

THESIS

Submitted in partial fulfillment
of the requirements for the degree of
DOCTOR OF PHILOSOPHY

by

AVINASH GAUTAM

Under the Supervision of

Prof. Sudeept Mohan

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

2016

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

PILANI (Rajasthan) India

CERTIFICATE

This is to certify that the thesis entitled "**Design and Implementation of Efficient Coordination Algorithms for Multi-Robot Systems**" and submitted by **Mr. Avinash Gautam,** ID No. **2009PHXF436P**, for the award of Ph.D. degree of the Institute embodies the original work done by him under my supervision.

Signature of the Supervisor

Name in capital letters: SUDEEPT MOHAN

Designation: Professor

Date: August 30, 2016

Dedicated

To God

For gifting me the most loving and caring parents, wonderful siblings, a lovely wife, and a beautiful daughter

ACKNOWLEDGMENTS

First and foremost, I extend my most earnest gratitude to my supervisor, Dr Sudeept Mohan, who has sustained me throughout my thesis with his patience and knowledge while at the same time affording me a way to act in my own style. Without his encouragement and effort this thesis would not have been completed or written. One simply could not wish for a better or friendlier supervisor.

In my daily work, I have been blessed with a friendly and cheerful group of fellow students who have helped in setting up a new mobile robotics laboratory in the department. Apoorv Umang, Prerak Mall, Krishna Murthy, Gourav Kumar, Arjun Ram, and Bhargav Jha deserve special mention.

I would like to offer my deepest gratitude towards Prof. J P Misra, Chief IPC and Dr. V S Shekhawat who have provided me with critical inputs continuously during the entire course of my PhD work. Many thanks to Prof. Bijay K Rout of the Department of Mechanical Engineering, for willingly sharing his knowledge and experience in robotics with me.

I am also indebted to Prof. S. Balasubramaniam, Prof. Navneet Goel, and Prof. Poonam Goel for their constructive comments during my end semester research presentations.

Thanks are due to Prof. Rahul Banerjee, HoD, Computer Science & Information Systems, for his help in various administrative issues in setting up a new mobile robotics laboratory in the department and making it possible for us to conduct this research.

Thanks are due to Prof. Souvik Bhattacharyya, Vice-chancellor and Prof. A K Sarkar, Director, BITS Pilani (Pilani campus) for their constant support and concern. I would wish to gratefully acknowledge Prof. S K Verma, Dean, ARD for the support.

The love and support I have received from my wife Dolly has allowed to me to work even harder for the culmination of this thesis. She has been there with me in every walk of life. It is significant to note the contributions of my little daughter Aaditri who is today 2.5 years old. She has played a role of stress reliever.

I would also like to thank my elder sister Anshu and younger brother Shashank for their understanding regarding my unavailability on many occasions.

I cannot conclude this acknowledgement without mentioning my parents. It would not have been possible to complete this thesis without their love, encouragement, support and blessings. There is no analogue of the sacrifices my parents have made for me in life.

Date: Signature: _____

Place: Name: Avinash Gautam

ABSTRACT

In multi-robot systems, individual robots cooperate with each other and work as a team to solve complex problems that are otherwise difficult to solve for a single mobile robot. There are several motivations for developing multi-robot solutions. Some of them are:

1. Some tasks are complex in nature and therefore it is difficult for a single robot to accomplish them, for example, parallel and simultaneous transportation of load.
2. Some tasks are monotonous, inherently parallel, and hazardous, for example, floor cleaning, lawn mowing, field harvesting, patrolling and battle field surveillance.
3. It is easier to build many simple robots rather than building a single monolithic omnipotent robot.
4. The use of multiple robots increases the robustness of the system i.e., makes the system more fault tolerant.

In this thesis we have proposed and implemented coordination algorithms for multi-robot systems for solving problem like, *geometric pattern formation*, *online terrain coverage* and *load balanced task decomposition and allocation*. We have shown that a properly coordinated robot team achieves better performance by efficiently utilizing the available system resources i.e. the robots.

The first problem investigated in this thesis is that of geometric pattern formation using multiple mobile robots. Specifically, the uniform circle formation problem has received considerable attention. Many researchers have addressed this problem from computational perspective and have suggested algorithms while treating the robots as abstract computational entities. We have designed and implemented a decentralized algorithm, referred to as STATE, for uniform circle formation using multiple mobile robots. One of the benchmark algorithm proposed by Défago and Konagaya is re-implemented on our multi-robot test-bed. The STATE algorithm is shown to perform better than the Défago and Konagaya's algorithm. The better performance of the STATE algorithm is due to its order preserving scheduling policy for multi-robot synchronization. On the other hand, the Défago and Konagaya's algorithm uses probabilistic scheduling policy for multi-robot synchronization which results in more number of activation steps and increased time for convergence of the algorithm.

The second problem investigated in this thesis is of terrain coverage. Terrain coverage has wide applicability in our day to day lives, ranging from small scale tasks like floor cleaning, lawn mowing, and harvesting to large scale missions like inspection of hazardous terrains and battlefield surveillance. These tasks are monotonous, time consuming and sometimes life-threatening. The problem of online terrain coverage (wherein the map of the environment is not known beforehand) using multiple mobile robots is investigated in this thesis. A centralized algorithm, Cluster, Allocate, Cover (CAC), based on frontier propagation is proposed. The CAC algorithm creates clusters of known frontier cells and allocates them to individual robots using an optimal allocation algorithm. One important characteristics of CAC is that it allows the robots to exploit the knowledge locally generated by them to its fullest and re-planning is done by the central controller only after each robot finishes its assigned task. Besides, several advantageous offered by CAC, it has two limitations i.e., it is not robust to failures and it does not cover the terrain contiguously. Another approach proposed in this thesis, referred to as FAST, is completely distributed, and is robust to failure of multiple robots. The robots executing FAST adopt a structured trajectory while covering the unknown region in a mutually exclusive manner which results in large unbroken and contiguous areas to be covered. Both the approaches, CAC and FAST, have been simulated and implemented on a multi-robot test-bed. Many other algorithms that are representative of the state-of-the-art have also been implemented and validated both in simulation and on a multi-robot test-bed. Empirical results obtained have shown that the two approaches suggested in this thesis perform better than many state-of-the-art approaches.

The third problem that is addressed in this thesis is of balanced partitioning of the unknown region. Many times multi-robot systems are deployed in an unknown environment wherein they have no prior knowledge of the tasks, for example, exploration of an unknown region, foraging, landmine detection etc. Two primary methods used for decomposing and representing the environment into segments are (a) grid based decomposition and (b) polygonal decomposition. Polygonal decomposition is shown to improve the efficiency of the multi-robot system as it minimizes the interference between the robots. If the partitions/segments of the environment are decomposed in a balanced manner, such that, the difference between the areas of the smallest and the largest partitions/segments is minimum and then the partitions/segments are assigned to the robots for processing, it will certainly improve the efficiency of the multi-robot system in solving the assigned tasks. In this thesis we have proposed an approach for balanced task partitioning which can be used in an

unknown indoor environment. It allows the robots to first discover the skeleton of the unknown region and then segment out elementary units like, rooms, halls, and corridors. The environment is then represented as a topological graph wherein each vertex of the graph corresponds to a particular elementary unit. This graph is then partitioned into maximally balanced and connected partitions with the application of genetic algorithms. The suggested approach is implemented in simulation and is compared against one of the most recent state-of-the-art approach. It is shown that the proposed approach creates better balanced partitions of the environment.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

MRS	Multi-Robot Systems
BSA	Backtracking Spiral Coverage
BSA-CM	Backtracking Spiral Coverage – Coordinated Multi-Robot
BSA-CMI	Backtracking Spiral Coverage – Coordinated Multi-Robot Improved
CAC	Cluster, Allocate, Cover
CM	Cost Matrix
CSC	Current System Configuration
FAST	Frontier Allocation Synchronized by Token Passing
HLD	High Level Design
IA	Instantaneous Assignment
LCMW	Look-Compute-Move-Wait
MANET	Mobile Adhoc Networks
MBCP	Maximally Balanced Connected Partitioning
MR	Multiple Robot Tasks
MRTA	Multi-Robot Task Allocation
MT	Multiple Task Robots
NDN	Nearest Downstream Neighbor
NSB	Null Space Based Behavior Control
NUN	Nearest Upstream Neighbor
PST	Partial Spanning Trees
RTT	Round Trip Time
SEC	Smallest Enclosing Circle
SLAM	Simultaneous Localization and Mapping
SR	Single Robot Tasks
ST	Single Task Robots
SYm	Suzuki-Yamashita model
UCF	Uniform Circle Formation
UGV	Unmanned Ground Vehicle
WLCM	Wait-Look-Compute-Move
WSN	Wireless Sensor Network

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

During the past two decades considerable attention has been paid to multi robot systems (*MRS*) which act as a team to accomplish some complex task. The interest of research community in this direction is justified because of the several advantages offered by *MRS*. The main motivation behind employing *MRS* is that they can increase the efficiency and effectiveness of the solution for a given task, i.e., a team of robots can perform a task in a superior manner by taking advantage of distributed sensing and actuation [Hungerford 2014, Pini 2011, and Mataric 1995a]. *MRS* can be classified into different categories based on their applications, for e.g., multiple industrial manipulators, military vehicles, or unmanned networked transport vehicles etc. In this thesis the term *MRS* refers to a team of mobile robots deployed in an indoor environment which are required to cooperate with each other in order to solve complex real world problems like, geometric pattern formation, simultaneous coverage of an unknown region, task allocation etc.

Cooperation in *MRS* has been extensively discussed in the literature and diverse definitions have been proposed. Some of them are:

(a) The first definition [Premvuti 1990] says that, cooperation is, *joining together for doing something that creates a progressive result such as increasing performance or saving time*. This definition aims at efficient utilization of the system resources.

(b) The second definition [Barnes 1991] says that, cooperation is *a joint collaborative behavior that is directed toward some goal in which there is a common interest or reward*. This definition is quantitative and is intended for profit maximization.

(c) In contrast to the two definitions given above, the third definition [Mataric 1997] says that cooperation is *a form of interaction, usually based on communication*.

Majority of the research in multi-robot task allocation (*MRTA*) is motivated by the first definition. The second definition leads to performance measurements in terms optimal completion time for a given task, by minimizing system resource utilizations. The last and the third definition encompass exchange of information about the robot's position (pose), and

belief (perception of the environment or map) with its peers. These definitions touch different aspects of cooperation in *MRS* i.e., the task, the performance of the robot team, and the communication protocol used.

Central to any multi-robot solution is an algorithm which decomposes a bigger task into elementary sub-tasks (task-decomposition) that are easier to solve by individual robots. These elementary sub-tasks are intelligently assigned to multiple robots so that they can solve these tasks in parallel with each other. Most of the times the robots work in a mutually exclusive manner, but, sometimes the sub-tasks overlap and the robots are required to negotiate and resolve conflicts. In the end these sub-tasks are recombined and the final results are obtained. This recombination also defines the overall behavior of the system i.e., answers how the termination condition is defined.

In multi-robot systems it is not only the performance of an individual but also the performance of the team which needs to be measured objectively. Key metrics on which the system performance can be measured are: task completion time, computational complexity of the algorithm, robustness/ completeness and fault tolerance. The performance of the robot team is dependent on the *task definition, group architecture of the system* (if it is centralized, weakly centralized, or distributed), *composition of the team* (if the multi-robot team is homogenous or heterogeneous), and the *communication structure* (ability of a given robot to recognize and model the intentions, beliefs, actions, and capabilities of other robots). The primary objective of *MRS* is to achieve better solution compared to a single complex robot for a given complex task. The existence of a better solution is largely dependent on the coherent and cooperative behavior of the robots. In order to achieve this objective individual robots in *MRS* communicate with each other. This communication can be *explicit* - by way of message passing or *implicit* - by way of sensing the features of the environment and localizing other robots.

A multitude of these aspects which are required to be addressed simultaneously makes it difficult to design and deploy multi-robot solutions in the real world. Nevertheless, researchers have developed solutions addressing the chief characteristics of *MRS* like, group structure, communication structure, control, group composition, learning, conflict resolution mechanisms, in an integrated manner. The interplay and strong correlation in the characteristics of multi-robot systems makes it difficult to bring about a classification which is focused on any single characteristic.

A classification based on different aspects such as size of the team, communication structure, self-organizing capability of the team, computational capability, and group composition is presented in [Dudek 1996] and is shown in Figure 1.1 below.

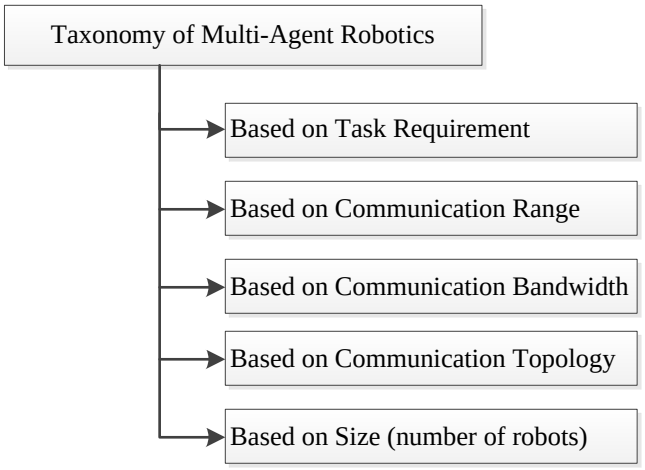


Figure 1.1 MRS Taxonomy [Dudek 1996]

Taxonomy of multi-robot systems based on coordination is presented in [Farinelli 2004] and is shown in Figure 1.2. In this classification the authors have described four different levels i.e., cooperation, knowledge, coordination, and organization.

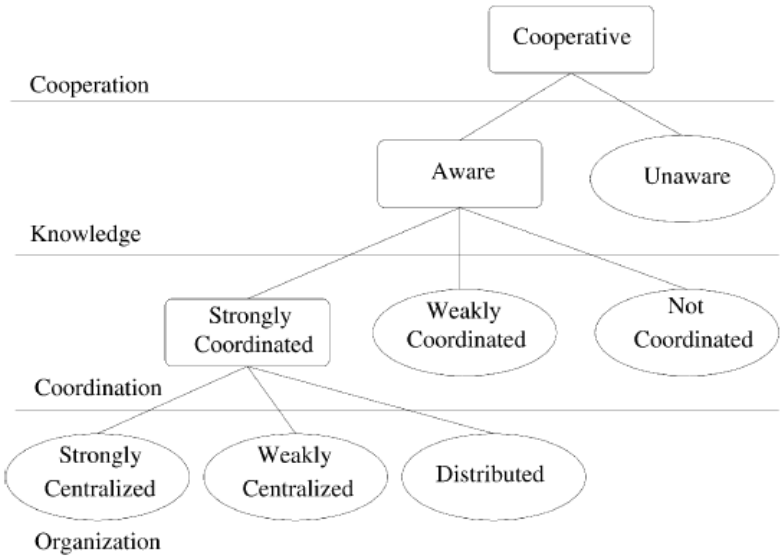


Figure 1.2 MRS Taxonomy [Farinelli 2004]

At the topmost level cooperative approaches are distinguished from non-cooperative approaches. At the knowledge level the robots might be aware of the presence of other robots in the environment or they may be completely unaware of the presence of other robots. At the coordination level multi-robot systems can be strongly coordinated, weakly coordinated and uncoordinated. In a strongly coordinated system the robots are more

involved and they deliberate by exchanging information. This information exchange certainly improves the efficiency of the task completion. In a weakly coordinated system, the robots are less involved and they communicate within the groups in an ad-hoc manner which can minimize the volume of communication. If the robots are aware and not coordinated, then it is the case of emergent cooperation in which the intelligent behavior is expected to emerge when all the robots perform a set of predefined actions against different stimulus from the environment. The organization level is concerned with the autonomy of individual robots to take decisions about what actions to perform. In a centralized organization, most often, a leader is elected who is the in-charge and is responsible for all the robots. Contrary to that in a distributed organization robots have complete autonomy and every robot in the team is free to decide their own actions. This classification of organization is further refined and weakly centralized organization has also been proposed wherein leaders may be re-elected if the currently elected leader dies or voluntarily resigns from its leadership position. Also the individual robots are partially autonomous.

Another classification was presented in [Gerkey 2004] is based on *MRTA* and coordination mechanisms. A domain independent taxonomy of *MRTA* problem is presented in this work. By "tasks" the authors mean "sub-goals" that are necessary to achieve for achieving the global objective. It is important for the designer of multi-robot system to understand the type of task allocation problem represented by the given task. For the same reason three axes for classification are defined as shown in Figure 1.3.

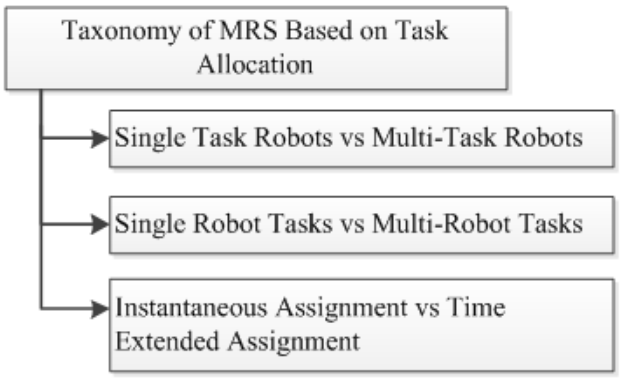


Figure 1.3 MRS taxonomy [Gerkey 2004]

Single task robots (*ST*) are capable of executing only one task at a time while *multi-task robots* (*MT*) can execute more than one task simultaneously at a time. *Single robot tasks* (*SR*) require only one robot to solve them while *multi-robot tasks* (*MR*) are to be solved by more than one robot. Instantaneous assignment (*IA*) means only one action or move ahead in

time can be planned. Time extended assignment means that the robots look ahead in time and the information available is sufficient to plan for future – more than one action or move ahead in time. The information is mostly related to the capabilities of the robots (sensors and actuators), specification of the tasks (what robot capabilities are required to accomplish the task, the number and location of the tasks, the arrival pattern of tasks), and the characteristics of the environment (location and geometry of the obstacles).

The literature on multi-robot systems encompasses many aspects, viz. path planning, communication, control, learning and adaptation, self-organization, simultaneous localization and mapping (*SLAM*) etc. In this chapter, the various issues concerning the development and deployment of solutions based on *MRS* are presented. The rest of this chapter gives a brief overview of *MRS*. Advantages of *MRS* are presented in Section 1.2. Different real-world applications and some aspects of *MRS* are presented in Section 1.3. The objectives of the thesis are presented in Section 1.4. The scope of this thesis is discussed in Section 1.5. Finally, the structure of the thesis is discussed in Section 1.6.

1.2 ADVANTAGES OF MULTI-ROBOT SYSTEMS

It has already been mentioned previously that carefully designed solutions based on multi-robot systems perform better than single robots. More complex tasks are now solvable with the advancement of robotic systems. Different aspects of multi-robot systems are discussed in this section and it is shown why multi-robot systems are favored over single robot systems.

- **Improved robustness**: The performance of multi-robot system should not suffer from failure of few robots. Since in a MRS there are multiple robots with similar capabilities, the redundancy offered by the system is exploited to compensate for the failure of few robots [Gautam 2016b, Yang 2011, Fazli 2010, Mead 2009, and Parker 1995]. Other robots should take charge of the task of the failed robot(s). It is true that performance of the system suffers from robot failures but it is always better to complete the task in some more time rather than not completing it at all. This is not the case in purely centralized systems where one robot is in charge of the whole team and hence there exists a single point of failure. The whole system stops working if this robot fails. Semi centralized systems supporting leader re-election or completely decentralized systems where every robot plans for itself are more preferred.

- **Improved efficiency**: Online terrain coverage, mapping and exploration, searching for an object of interest in an unknown region are different tasks which can be completed faster as the robots can quickly disperse in the environment.
- **Parallelism**: Multiple robots can execute many tasks in parallel thus reducing the task completion time. With a single robot multiple tasks can be completed only in a sequential manner.
- **Reduced cost**: Design and construction cost of many simple robots is much lesser than that of one monolithic robot equipped with different kinds of costly sensors and actuators.

These aspects impress upon us to adopt multi-robot solutions but it certainly does not imply that multi-robot systems are easier to control, manage or monitor when compared to single-robot systems. There are several challenging problems in MRS which do not exist with single robots and cannot be ignored. For example, localization of peer robots, communication for information exchange, and synchronization for solving overlapping tasks etc. Only careful coordination of multiple mobile robots delivers the benefits offered by harnessing the true potential of multi-robot system. If it is a centralized system, then a single task can be easily decomposed and optimally assigned to each robot, enabling multiple robots to solve the sub-tasks in a mutually exclusive manner without interference. On the other hand, in a decentralized system the following question must be answered (not all of these are however applicable in all situations):

- How to decompose the bigger task into smaller (elementary) sub-tasks good enough for a single robot to accomplish?
- Once the bigger task is broken down into several elementary sub-tasks, how the elementary tasks will be assigned to individual robots?
- What kind of communication structure should be implemented between the robots? If there is a dependency between the two tasks one robot might have to wait for the other to finish its task and make the results available. Both robots should be aware of this dependency and inform each other of the status of their own tasks. This can only be achieved with the help of communication.

1.3 APPLICATIONS AND ISSUES OF MULTI-ROBOT SYSTEMS

Cooperative robotics or multi-robot systems is a young and dynamically growing field of research. In the last two decades several solutions based on *MRS* have been developed which has made it possible to classify these systems based on the type of the robots and the context in which they are used. There are mobile robots which are ground vehicles and are extensively used in both indoor [Turtlebot 2016, Lizi 2016] and outdoor environments [Husky UGV 2016]. Other types of robots have also been used in different applications. For example, under water autonomous vehicles [Heron USV 2016] are used in sea exploration and unmanned aerial vehicles [Parrot ARdrone-2.0 2016] are frequently used for surveillance, mapping, etc. Multi-robot systems can be employed in different fields for e.g., industrial, defense, biomimetics (imitation of the models, systems, and elements of nature for the purpose of solving complex human problems) etc. Some of the applications of multi-robot systems include terrain coverage, area exploration, parallel and simultaneous transportations of load etc. To be able to successfully accomplish these missions the multi robot solution demands intelligent and strong coordination between individual units. This means that the robots have to exchange information with their team mates about their relative positions and of their current belief about the environment and its features. In [Mataric 1995b] some empirical results have been presented for a box pushing experiment using two legged robots. A team of differential drive robots is shown to perform box pushing in [Yamada 2001] with no communication between the robots. An object handling task is accomplished in a cooperative manner in [ZhiDong 2003]. Robot motion planning based approach wherein multiple robots are employed to transport bigger objects is suggested in [Yamashita 2003]. In [Tanner 2003] the authors have used two robots with manipulators to carry a deformed object.

Terrain coverage and area exploration are two significant tasks for which *MRS* are used. Terrain coverage requires every unobstructed region (free space) to be visited by at-least one robot whereas area exploration requires the robots to jointly build the map of an unknown environment. For terrain coverage several algorithms exist for both known and unknown maps. When the map of the terrain is known it is termed as offline coverage [Agmon 2008]. When the map of the terrain is not known a priori (online coverage) then the task becomes complex and requires the robots to build the map in an incremental manner in order to visit the free space. To achieve their mission of coverage, the robots are required to strategically coordinate with each other, explore different parts of the terrain and then merge the results to

build a global map. This follows divide and conquer strategy which results in efficient coverage in less time. Online coverage of the terrain is a non-trivial task which requires the robots to localize themselves and their peers with respect to each other and the features present in the environment. Simultaneous localization and mapping (*SLAM*) has been one of the frontier areas of research in robotics. A rigorous discussion on single robot *SLAM* is presented in [Durrant-Whyte 2006, Bailey 2006]. Variety of solutions have been proposed for multi-robot *SLAM* for e.g., in [Rekleitis 2001] the authors have shown the robots exploring and building maps of vast environment while localizing themselves with respect to each other and reducing odometric errors. A greedy approach for map building and exploration using *MRS*, based on market economy is presented in [Zlot 2002] wherein the robots try to maximize their individual profits by minimizing the cost of travelling to frontiers and maximizing the information gain on reaching those frontiers. Considerable research work in cooperative robotics has been directed to analyze and understand biological systems. With the introduction of behavior based control paradigm [Arkin 1998], robotics researchers inspired from the social characteristics of insects and animals, began developing engineering solutions which imitate biological swarms. It is very interesting to observe such analogies for e.g., fighter aircraft and naval formations imitating bird flocks [Krontiris 2011], as shown in Figure 1.4.

(a) (b)

Figure 1.4 (a) A bird flock in a wedge like formation. (b) Indian air force squadron flying in a wedge like formation

In *MRS* this is achieved by applying primitive control laws governing the colonies of living organisms like ant colonies, fish schools, animal herds etc. Several intelligent behaviors for e.g., foraging [Balch 1999, Shell 2006], flocking [Xiong 2010], dispersion [Damer 2006], have been reproduced by *MRS*. Flocking is one of the most important form of coordination

in multi-robot systems wherein the robots tend to move in a formation while preserving a pattern. The robots flock together to achieve some common group objective for e.g. patrolling, surveillance, delivering payloads etc. In 1987, Reynolds introduced three heuristic rules which lead to the creation of the first computer animation of flocking [Reynolds 1987]. For a successful flock he suggested the following three rules:

(a) *Flock centering:* individuals attempt to stay close to their flock mates.

(b) *Obstacle avoidance*: individuals avoid collision with nearby flock mates.

(c) *Velocity matching*: individuals attempt to match their velocities with nearby flock mates.

In the literature these rules are also termed as *cohesion*, *separation*, and *alignment*. These rules were abstract and had very broad interpretations. It could only become possible to correctly interpret these rules when Reynolds published more follow up papers [Reynolds 1999] which describe steering behaviors of autonomous characters in computer animation and which introduces a method for constructing large groups of autonomous characters [Reynolds 2000]. These autonomous characters were then made to respond in real-time and interact with the user, as well as with other characters and the environment. The remarkable works of Reynolds initiated extensive research on the group behaviors and dynamics [Mataric 1992, Parker 1993, Kube 1993] and was followed by further research on multi-robot pattern formations [Balch 1998, Wang 1991].

Two more applications that are closely related to multi-robot systems are control of wireless sensor networks (WSNs) and mobile ad-hoc networks (MANETs). Particularly, MANETs require autonomous nodes to interact with each other in a distributed fashion i.e., without using any fixed base station or network infrastructure. MANET nodes can communicate with other nodes which are in their vicinity which is defined by their transmission range. Distant nodes are reached by relaying packets over a multi-hop network comprising of intermediate nodes using store and forward mechanism. MANET based approaches have been investigated by researchers for multi-robot search and rescue missions and exploration of an unknown regions. The flexible and the fault tolerant communication capability offered by MANETs has attracted researchers to propose MANET based multi-robot communication [Wang 2005, Sit 2007, Witkowski 2008]. Multi-robot systems have also been employed for variety of tasks in the military research for e.g., battle field surveillance [Trebi-Ollennu 1999], area patrolling [Marino 2013], and intrusion detection [Fagiolini 2008, Fagiolini 2009]. Several research efforts are also visible for multi-robot soccer playing (soccer robots) in competitions like robocups [Candea 2001, Weitzenfeld 2006b] etc.

1.4 OBJECTIVES OF THE THESIS

The main objective of the thesis is to design efficient coordination algorithms for solving different real-world problems using multi-robot systems. The three domains chosen for this investigation are geometric pattern formation, online terrain coverage and area decomposition/ partitioning. Different objectives of the thesis are classified according to these three problem domains. Following is the detailed list of objectives.

1. **Geometric pattern formation**

   (a) Study and identify the limitations of available solutions.

   (b) Design efficient decentralized algorithm(s) for solving the uniform circle formation problem.

   (c) Establish an experimental test-bed comprising of mobile robots for testing the efficacy of the algorithm(s) proposed in this thesis.

   (d) Re-implement some of the state of the art approaches for the purpose of comparison with the algorithm(s) proposed in this thesis.

2. **Online Terrain Coverage**

   (a) Study and identify the limitations of available solutions.

   (b) Design and simulate both centralized and decentralized algorithm(s) that are efficient in terms of reduced coverage completion time and reduced overlapping coverage when compared with some of the state of the art approaches.

   (c) Establish an experimental test-bed comprising of mobile robots for testing the efficacy of the algorithm(s) proposed in this thesis.

   (d) Re-implement some of the state of the art approaches on the established test-bed for the purpose of comparison with the algorithm(s) proposed in this thesis.

3. **Area Partitioning/ decomposition**

   (a) Study and identify the limitations of approaches for polygonal partitioning.

   (b) Design and simulate algorithm(s) that are efficient in terms of balancing the size of the partitions, such that, the area to be processed by individual robots is almost equal.

   (c) Re-implement some of the state of the art approaches in simulation for the purpose of comparison with the algorithm(s) proposed in this thesis.

1.5 SCOPE OF THE THESIS

Many requirements are imposed on multi-robot systems in literature. Not all the requirements have been addressed in this thesis. The scope of the thesis in terms of these requirements is summarized as follows:

(a) **Fault-tolerance**: The multi-robot system should be robust to failures of robots. In other words, there should not be a single point of failure. Centralized or leader-follower approaches are incapable of handling failures. In this thesis, we have suggested decentralized algorithms for geometric pattern formation and fault-tolerant algorithm for online terrain coverage. The scope of thesis is limited to handling robot failures alone. Byzantine failures and communication failures are not in the scope of the thesis.

(b) **Speed**: The multi-robot system should complete the assigned task quickly in an efficient manner. The suggested algorithms in this thesis have addressed this requirement and are demonstrated to achieve better results than some of the state of the art approaches.

(c) **Task allocation**: The multi-robot systems should be able to decompose a bigger (global) task into smaller task subsets and then optimize the allocation of task subsets to individual robots in a decentralized manner. The task allocation strategy should ensure efficient and faster completion of the global task. This requirement is carefully addressed in the thesis.

(d) **Scalability**: The multi-robot system should easily be able to accommodate the addition/ subtraction of robots during operation. This requirement is addressed in a limited sense in the thesis. The algorithm proposed in this thesis for solving the uniform circle formation problem allows addition and removal of robots once the pattern is established. The fault-tolerant decentralized algorithm proposed in this thesis for online terrain coverage allows the robots to be added and removed and it has self-organizing nature. The multi-robot system considered in this thesis is inspired from higher order animal species like wolf-pack hunting model [Weitzenfeld 2006a] and humans (robo-soccer playing experiments [Weitzenfeld 2006b]), therefore, the multi-robot team size is restricted to ten robots. This thesis does not address the problems related to swarm robotics wherein the multi-robot team size of some hundreds or thousands is considered.

(e) **Implementation**: The multi-robot system should be implemented and proven on physical system. In this thesis, most of the work has been experimentally validated and tested on a team of mobile robots. Extensive efforts have been put in establishing and realizing different centralized and decentralized algorithms on a multi-robot test-bed. Most of the experiments however are conducted in a controlled laboratory setting.

Some other requirements that are not addressed at present in this thesis and may be addressed in future are:

- *Communication*: The multi-robot system should be capable of dealing with limited and imperfect communication.
- *Extensibility*: The multi-robot system should be easily extendable to accommodate new functionalities.
- *Heterogeneity*: The multi-robot system should have ability to accommodate heterogeneous teams of robots.
- *Learning*: The multi-robot system should be able to adapt itself for specific applications in real-time (on-line).

1.6 STRUCTURE OF THE THESIS

A brief overview of the research work carried out in the individual chapters is presented below:

Chapter 2: Literature Review

This chapter presents a review of the related works in this field of multi-robot systems conducted by researchers, including works on geometric pattern formation, online terrain coverage and area partitioning/ decomposition.

Chapter 3: Geometric Pattern Formation by Multiple Mobile Robots

Multi robot pattern formation has a wide range of applications like inspection of hazardous regions, parallel and simultaneous transportation of load, area exploration, etc. This problem has been investigated both from the scientific and engineering perspectives. There is a whole body of experimental work on robot formations and at the same time there is no dearth of theoretical research addressing this problem. Several assumptions considered in these theoretical studies are way too simplified. It is assumed that these assumptions will

somehow be reasonably approximated. A practical model is suggested in this chapter for geometric pattern formation. This model uses approximate solutions to some of the assumptions considered in theoretical research works. A decentralized algorithm (STATE) is proposed in this thesis and is shown to perform better than one of the benchmark approach i.e. the Défago and Konagaya's algorithm [Défago 2002, Défago 2008] for uniform circle formation. Both the algorithms are implemented on a multi-robot test bed. An independent software system is developed for localizing multiple robots moving on a floor in an indoor environment. The proposed system for multi-robot localization employs image processing algorithms to read the markers attached on top of the robots and determine their pose using an overhead camera. A framework for inter-robot communication has been developed for asynchronous and non-blocking robot-to-computer, and robot-to-robot communication.

Chapter 4: An Efficient Approach for Online Multi-Robot Terrain Coverage

In this chapter, an algorithm for online multi-robot coverage is suggested. The suggested algorithm is a centralized algorithm that proceeds with minimal knowledge of the already explored region and the frontier cells. It creates clusters of frontier cells which are allotted to robots using an optimal assignment scheme. Coverage is then performed by each robot using a modified motion plan which results in less redundant movement. Some approaches use data clustering algorithms like *K*-means for environment decomposition. These approaches do not specify strict time criteria for re-clustering. Moreover, the motion plans they use result in redundant coverage. To overcome these limitations, an appropriate motion plan for the robots is chosen based on the context of already covered frontiers. Dispersion of robots is an emergent behavior in this approach. The efficacy of the proposed approach is tested in simulation and on a multi-robot test-bed.

Chapter 5: Synchronous Frontier Allocation for Scalable Online Multi-Robot Terrain Coverage

In this chapter, we have proposed, **F**rontier **A**llocation **S**ynchronized by **T**oken passing (FAST), a distributed algorithm for online terrain coverage using multiple mobile robots, which ensures mutually exclusive selection of frontier cells. Many existing approaches cover the terrain in an irregular fashion, without considering the usability of the already covered region. For instance, in the task of floor cleaning in an office building, these approaches do not guarantee the cleanliness of large unbroken areas until a majority of the task is complete. FAST on the other hand, incrementally traverses the terrain generating structured trajectories

for each robot. Following a structured trajectory for coverage path planning is proven to be a powerful approach in literature. This renders large portions of the terrain usable even before the completion of the coverage task. The proposed map representation techniques used in FAST render it scalable to large terrains, without affecting the volume of communication among robots. Moreover, the distributed nature of FAST allows incorporation of fault-tolerance mechanisms. Empirical investigations on maps of varied complexities and sizes both in simulation and on an experimental test-bed are discussed in this chapter.

Chapter 6: Balanced Partitioning of an Unknown Region for Efficient Multi-Robot Coordination

We consider the task of partitioning of an unknown region into sub-regions which are then apportioned to a multi-robot system for further processing. It is evident from the literature that polygonal decomposition of workspace is superior than grid based decomposition. Many existing approaches are based on Voronoi partitioning. However, they produce unbalanced partitions resulting in uneven distribution of the workload to individual robots. This chapter proposes an approach to create balanced partitions of the unknown region so that the regions to be processed by individual robots are almost equal in area. This type of partitioning can be especially useful in applications like floor cleaning, surveillance, and patrolling, wherein fair distribution of the workload between the agents is important. The proposed method requires the robots to use minimalistic sensors to first determine the boundaries of the unknown region which is then converted into a weighted connected graph. Further, this graph is partitioned into sub-graphs which are maximally balanced using genetic algorithm. These sub-graphs can be optimally allocated to the available robots. To the best of our knowledge this is the first attempt in this direction.

Chapter 7: Conclusion and Future Work

This chapter concludes the thesis by summarizing the research work presented in the previous chapters and the scope for the future work. It also enumerates the publications done based on the research work in this thesis.

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

Multi-robot system research has gained considerable attention during the past two decades. In the following section, a detailed literature review on the coordination algorithms for multi-robot systems has been presented. In particular, a review of multi-robot solutions suggested in three different application domains namely, geometric pattern formation, online terrain coverage, and area partitioning/decomposition is presented. Specific description of the three aforementioned application domains follows:

(a) *Geometric pattern formation*: Robot formations play a vital role in many applications and can be more advantageous than deploying a single mobile robot. For instance, robot formations can act as sensor arrays when organized in a lattice. This arrangement allows them to collect and exchange rich information about the state of their environment. Robot formations can be of great use when building the map of an unknown region. In [Howard 2004], the authors have exploited the redundancy in the measurements of multiple robots regarding the state of the environment to build an accurate and better quality map. Another advantage of geometric pattern formation is that, when the formation is established, individual robots can be assigned or they may assume different roles in task execution. The geometric pattern formation problem can be defined as a problem wherein a group of mobile robots are required to establish a predetermined geometric shape. It starts with an assumption that the shape is not known in advance to the robots; only information that is available of the robots is the relative positions of their peers. The geometric pattern formation is a problem that demands a distributed algorithm which is independently executed by each robot. The algorithm must ensure that the robot team converge to a particular geometric shape/formation in some finite iterations. In that sense, considerable work has been done on uniform circle formation problem and it has become a benchmark for comparing pattern formation algorithms. The problem of uniform circle formation is one of the focus of this thesis.

(b) *Online terrain coverage*: The problem of distributed coverage of an unknown environment has emerged as a very important problem in multi-robot research. Online

terrain coverage requires the team of mobile robots to physically traverse the entire free space (unobstructed region) of the unknown environment while reducing the coverage completion time and the amount of overlapping or redundant coverage. Both centralized [Kapanoglu 2012] and decentralized [Andries 2013] algorithms have been suggested in the literature. These works are carried out both in simulation and/or validated on an experimental test-bed in a laboratory setting. Variety of assumptions about the communication, sensing, processing and storage capabilities of the robots have been considered. For instance, some approaches consider limited communication range [Rekleitis 2004] while some other approaches consider limited sensing range/ visibility range [Fazli 2010]. There are approaches that consider a homogenous team of mobile robots doing coverage [Raghavan 2007] and some other approaches consider a team of heterogeneous mobile robots doing coverage [Yazıcıoğlu 2013]. Some approaches consider a team of low powered mobile robots with minimal processing [Cannata 2011] while on the other hand there are approaches that employ relatively powerful mobile robots [Ozkan 2010]. A large number of approaches suggested for online terrain coverage with different robot capabilities makes it difficult to compare and contrast them and therefore opens up opportunity for further research.

(c) *Area partitioning/ decomposition*: Autonomous mobile robots require a map of their environment for successful navigation and path planning. Therefore, they build the map of their environment which is primarily represented in the robot's memory in one of the two possible ways: grid based decomposition [Burgard 2005] and polygonal decomposition [Wurm 2008]. Polygonal decomposition of maps is proven to be better than grid based decomposition of the maps when it comes to automating tasks like exploration of an unknown region using multi-robot systems. Topological graphs are constructed on top of polygonal decompositions that considerably reduce the processing efforts for computing paths for mobile robot navigation. Spatial division of the map of the environment into meaningful elementary units has direct impact on the performance of the task [Bormann 2016]. Less attention has been paid towards measuring the impact of different area partitioning approaches. In particular, balancing the spatial partitions of the workspace is largely an untouched problem.

The rest of this chapter is organized into three sections. In Section 2.2 the problem of geometric pattern formation has been reviewed. Specifically, the problem of uniform circle formation and its available solutions has been discussed. Different approaches addressing the

problem of online terrain coverage have been reviewed in Section 2.3. Various methods of area partitioning/ decomposition have been reviewed in Section 2.4.

2.2 GEOMETRIC PATTERN FORMATION

Some of the significant research works on multi-robot pattern formation are reviewed in this section. Experimental research works are reviewed first in Section 2.2.1 followed by purely theoretical studies in Section 2.2.2.

2.2.1 Experimental Research on Multi-Robot Pattern Formation

In this section, some of the approaches that are representative of the experimental studies in multi-robot pattern formation are discussed in a chronological order. In [Fredslund 2002], using only local sensing and minimal communication the authors have been successful in bringing a group of distributed robots in a predefined geometric shape. All robots identify a single friend robot using a pan and tilt camera. The robots always try to keep the friend robot in the center of the cameras field of view. Each robot is assigned a unique ID. Initially, before the formation is established the robots remain in a chain of friendship which is sorted by IDs. It is a neighbor referenced approach (a robot positions itself with respect to a neighbor robot) and therefore the robots continuously keep polling the neighbor robots which results in a waste of resources. Unlike [Fredslund 2002], the algorithm suggested for uniform circular formation in Chapter-3 i.e. the STATE algorithm [Gautam 2016a] does not require the robots to continuously poll their neighbors, neither it demands that the robots should be positioned in a predefined initial formation. In [Fax 2002], a motion planning approach for formations of mobile robots is presented. In this approach, a central controller explicitly plans the trajectories for individual robots while they maintain a predefined formation. This is done to allow the formation to change dynamically while performing complex maneuvers in the presence of obstacles. Unlike [Fax 2002], the STATE algorithm [Gautam 2016a] is a decentralized approach i.e. each robot independently executes the algorithm for circle formation.

A notion of local templates is used in [Krishnanand 2005] which is similar to the concept of potential fields wherein the gradient of the field is followed by the robots. The robots successfully self-assemble into a grid, line, and wedge formations. But sometimes the robots may get trapped in local minima. This is an inherent limitation of the potential fields. In [Hao 2005], the authors have presented a practical framework for planning and control of formations of three UGVs. The main objective is to allow the formation to change in real-

time in order to avoid obstacles while navigating in a dynamic environment. In [Marshall 2006], the authors have conducted experimental implementation of multi-robot flocking and cyclic pursuit. The control laws are solely governed by local sensing and data processing. No use of explicit communication, GPS, or overhead camera is reported. Colored markers are attached to the robots which are tracked by other robots thus enabling them to continuously poll each other as they move. This involves complex image processing and doing it on onboard computer of the robot is an expensive operation particularly for applications like flocking. In [Antonelli 2007], two experiments have been conducted using a team of six mobile robots to determine the performance and robustness of a null space based behavioral (NSB) control method for formation control in multi-robot systems. The approach was implemented in a centralized manner for controlling a platoon of autonomous robots at kinematic level. The NSB has proved to be a powerful control mechanism. In [Antonelli 2009a], the authors have successfully conducted experiments on formation control for a team of grounded mobile robots in the presence of both static and dynamic obstacles. Further, the authors have conducted experiments on flocking behavior under different conditions (moving rendezvous, 2D and 3D space, and in the presence of obstacles) of multi-robot systems in [Antonelli 2009b].

The problem of pattern formation is addressed in [Mead 2009] for terrestrial robots. The formation is encoded in a cellular automaton. Individual members of the automaton are treated as an individual cell. An auction based approach is used for the recovery of formation in the event of failure. The robots are continuously required to poll their neighbors for a possible movement in order to align themselves correctly with respect to the desired pattern. This is not an efficient utilization of the system resources. Moreover, uniform circle formation is not possible with this approach. Leader-Follower based control architecture is suggested in [Monteiro 2010] so that a swarm of robots can be made to flock in a formation to a goal location. The goal location is known beforehand to the leader. Obvious disadvantages of centralized architecture also apply to this approach i.e., single point of failure. To this end we have seen some of the significant experimental research works carried on multi-robot pattern formation are control theoretic and provide solutions for flocking problem. In addition to these approaches a whole body of work on multi-robot pattern formation exists in theoretical computer science. In the next section we discuss some the significant theoretical research works.

2.2.2 Theoretical Research on Multi-Robot Pattern Formation

In this section, we discuss two well-known system models followed by a detailed discussion on different approaches suggested by researchers under these models.

A. Suzuki-Yamashita and CORDA model

There are two well-known system models, the Suzuki-Yamashita model [Suzuki 1999] (SYm) and the CORDA model [Prencipe 2001]. These two models are hypothetical and therefore the robots and their world model are treated as an abstraction. The basic assumptions of the two models are listed in Table 2.1.

Table 2.1 Comparison of the basic assumptions of SYm and CORDA models

Model	**SYm**	**CORDA**
World model	$\Re^2$	$\Re^2$
Dimension	Point robot	Point robot
Mobile	Yes	Yes
Autonomous	Yes	Yes
Anonymous	Yes	Yes
Visibility	Unlimited	Unlimited
Coordinate System	Local	Local
Agreement	Partial	Partial
Communication	Implicit	Implicit
Synchronization	Semi-synchronous	Asynchronous
Clock	Global clock	Not Applicable

The robots are considered as a unit having computational capabilities. The world model of the robot is a two dimensional plane ($\Re^2$). The robots are treated as a point mass. Each robot is mobile and can freely move in $\Re^2$. The robots are autonomous i.e. they carry out their computations independent of each other. The robots are anonymous in the sense that they are indistinguishable or do not have unique identities. In addition, the robots are assumed to have unlimited visibility i.e. they are equipped with sensors to instantaneously detect the position and heading of other robots. The robots construct their own local view of the world i.e. they see the other robots in their own local coordinate system and are egocentric. However, the robots are assumed to have partial agreement on the local coordinate system. In particular, the robots agree on the direction and orientation of one of the coordinate axis, say the *X* axis. They also agree on clockwise and anti-clockwise direction. The robots in the

two models cannot explicitly communicate and the only means of communication is by sensing other robots.

The main difference between the two models is in the level of synchronization i.e., the way the activation cycles of the robots are synchronized. In SYm, the activation cycle of a robot is a sequence of events Look-Compute-Move-Wait (LCMW), and in the CORDA model it is Wait-Look-Compute-Move (WLCM). It is to be noted that the time spent by the robots in each state of the activation cycle is not known or fixed, but it is finite. The SYm is a semi-synchronous model. A global clock is assumed and the time is represented as an infinite sequence of discrete time instances. The robots, during each time instance could be either active or inactive. It is also assumed that the robots always become active only at the beginning of the time step. It is assumed that the robots execute their activation cycle almost instantaneously (atomically). This assumption implies that the robots only sense other robots when they are stationary. Contrary to this the CORDA model is asynchronous and makes no assumption on the cycle time of the robots. As a consequence of this assumption, while moving itself a robot can see other moving robots. This however does not mean that a robot can distinguish between a moving and a stationary robot. This limitation renders it difficult to design algorithms for control and coordination of multiple mobile robots in CORDA and is one of the main reasons why SYm is more preferred. We now present a detailed literature review on theoretical research on geometric pattern formation in cooperative robotics. Specifically, more stress is given to uniform circle formation problem.

B. State of the Art

As an illustration of self-stabilizing distributed algorithms a brief discussion on the problem of circle formation is presented in [Debest 1995], however, no solution is provided. Different algorithms on various geometric pattern formations have been proposed in [Sugihara 1996]. The authors have proposed a heuristics based algorithm which produces an approximation of a circle i.e., a Reuleaux triangle is obtained. A non-oblivious algorithm for the formation of regular polygon is proposed in [Suzuki 1999]. Eventually by executing this algorithm the robots get arranged at regular intervals on the circumference of a circle. However, the robots need to remember all of their past states and actions. An algorithm for point gathering is proposed in [Ando 1999] under Suzuki-Yamashita model, that supports robots with limited visibility. The point gathering problem is solved in [Flocchini 2005] under the CORDA model. Here the assumptions on the atomicity of the activation cycle and instantaneous movement are dropped, but common sense of direction of the two coordinate axes is

considered. Based upon the common knowledge possessed by the multi-robot system the authors in [Flocchini 1999] have studied the solvability of arbitrary pattern formation problem. An extensive survey has been conducted in [Cao 1997] and the authors have advocated that only few research works address the geometric pattern formation problem from computational perspective. This observation is also reinforced in [Flocchini 2005].

Robot formations can act as sensor arrays while exploring and/or monitoring an unknown terrain. For area exploration of unknown terrain, circular formation tends to be the most suitable choice [Suzuki 1999, Krick 2009, Leonard 2010]. Many authors [Chatzigiannakis 2004, Défago 2002, Défago 2008] have generalized the semi-synchronous model i.e., SYm. An algorithm for uniform circle formation in the semi synchronous setting has been proposed in [Défago 2002]. The authors have decomposed the problem into two sub-problems which are then solved by them separately. The first sub-problem is to guide the robots to the circumference of a non-degenerated circle and the second sub-problem is to allow the robots to execute a deterministic algorithm that eventually brings them into a configuration where all the robots are spaced evenly i.e., for N robots the angular distance between any two direct neighbors is $2\pi/N$. An important characteristic of the solution to the second sub-problem is that the robots calculate and move half way distance towards the mid-point between the two direct neighbors. This is done so that the multi-robot system does not get trapped in a livelock. In this work the two direct neighbors of a robot can be active at the same time. A computer simulation of this work has been carried out in [Souissi 2004] wherein the robots are probabilistically activated and their convergence time in terms of total number of activation steps has been measured. The probabilistic activation schedule does not guarantee fairness and as a result, the algorithm takes more number of activation steps to converge. The approach suggested in [Défago 2002] is extended in [Défago 2008] and a more rigorous proof is provided.

In [Gautam 2012], a practical software framework which encapsulates and abstracts robot behaviors and allows the robots to execute a centralized algorithm for uniform circle formation is presented. Some of the best practices of software engineering and well known design patterns like observer and decorator have been employed. This increases the extensibility of the framework to accommodate newer algorithms for different applications of cooperative robotics. Within this framework, in [Gautam 2013b], the authors have suggested an algorithm for uniform circle formation based on the solution of the dining philosopher problem suggested in [Garg 2004]. This scheduling policy does not allow direct

neighbors of a robot to become active at the same time. It ensures fairness and the algorithm converges to a uniform circular formation. The work is mainly a simulation and no comparative study has been done. The algorithm suggested in [Gautam 2013b] was experimentally validated in [Gautam 2014]. The algorithm was executed on a single computer in a centralized manner and movement commands were sent to the robots in a synchronized manner. Moreover, the robot to computer communication was synchronous and blocking. In [Elor 2011], discrete time algorithms for uniform circle formation where the agents are allowed to move only in one direction on the vertices of a ring graph have been proposed. A similar approach wherein the agents move only in one dimensional continuous space of the circle has been suggested in [Flocchini 2008]. To this end it is seen that the pattern formation problem, more specifically the uniform circle formation problem, for weak robots, has been addressed widely by researchers from theoretical and scientific perspective [Elor 2011, Flocchini 2008, Dieudonné 2008, Dieudonné 2009].

At this point, we argue that not much attention has been paid to validating these algorithms on a real multi-robot test bed, nor an attempt has been made to compare the performance of these algorithms with other similar approaches. Therefore, it becomes important to understand how these assumptions can be approximated in order to implement and validate these algorithms on a physical multi-robot test-bed. The STATE algorithm [Gautam 2016a] suggested in Chapter 3 of this thesis is a distributed algorithm that allows a team of autonomous mobile robots to achieve a uniform circular formation. It is a real implementation of a distributed robotics system tested on a team of five e-puck robots. The robots communicate with each other by way of message passing. The robot to computer and robot to robot communication is absolutely asynchronous and non-blocking. Each robot individually runs the proposed algorithm on-board micro-controller (dsPIC) of the robot. One of the benchmark algorithms for uniform circle formation [Défago 2002, Défago 2008], referred to as DK algorithm, is also implemented on our experimental test-bed. Statistically it is found that the STATE algorithm performs better than the DK algorithm. Detailed explanation of the two algorithms, their implementations and the design of the experimental test-bed are discussed in Chapter 3 of this thesis.

2.3 ONLINE TERRAIN COVERAGE

Terrain coverage and area exploration are closely related problems. Online multi-robot exploration requires the robot team to explore and build the global map of an unknown region. Some exploration approaches [Wurm 2008, Puig 2011, Sheng 2006, Dasgupta 2009]

aim at reducing the redundancy in exploration and target global dispersion for speedy completion of the exploration task, whereas some other approaches [Mukhija 2010, Yamauchi 1998, Rooker 2007, Zlot 2002] target local dispersion and exploit the redundancy in exploration to improve the accuracy of the global map constructed, thus enabling the robots to localize better. Exploration does not require the robots to physically traverse the entire free space in the unknown region. On the other hand, online terrain coverage can be stated as a problem of finding trajectories for individual robots such that the entire free space in an unknown region is physically traversed at least once, by some robot(s). The primary objectives of terrain coverage are minimizing overlap in regions covered by different robots and minimizing the time taken to complete the coverage. These objectives are no different than those of exploration, with the exception that the map of the environment is no longer a deliverable. However, a map of the environment is still required by the robots to localize themselves and for generating non-overlapping coverage paths. Hence we argue that an exploration algorithm can be adapted for the purpose of terrain coverage by restricting the robot's sensing range within a unit distance which is equal to the footprint of the robot. Approximate cell decomposition of the search space is a popular method of coverage path planning and is reviewed in [Galceran 2013]. In [Andries 2013], approximate cell decomposition based methods for terrain coverage and area exploration are classified into three different categories, i.e. (a) Tree cover algorithms (b) Ant-based algorithms and (c) Frontier propagation algorithms.

In *tree cover algorithms*, a spanning tree is constructed over the environment, which is assumed to be known (i.e., offline terrain coverage). Segments of this spanning tree are then apportioned to multiple robots for coverage. Some examples of tree cover algorithms are [Hazon 2005, Zheng 2010, Senthilkumar 2012]. *Ant-based algorithms* are inspired from the pheromone laying and trail sensing nature of ants [Andries 2013, Ferranti 2007, Ferranti 2009]. These algorithms are shown to be completely distributed and robust. In these methods, localization and communication are achieved by treating the environment as a shared memory on which pheromone trails released and deposited by robots are read by other robots. Failure of one or more robots and/or addition of robots are gracefully dealt with. Some algorithms consider special substances for use as pheromones. For example, alcohol trails and ink markings have been used in [Sharpe 1998] and [Svennebring 2004] respectively. Others allow the robots to drop smart tags in the environment [Andries 2013, Ferranti 2007, Ferranti 2009]. These smart tags can communicate with each other and can

also be read and updated by the robots. Despite their advantages, ant-based algorithms are not widely used since they are hard to realize. For instance, the implementation of precise trail laying and sensing hardware turns out to be challenging and the smart tags themselves are not fail-safe. *Frontier propagation algorithms* have received considerable attention in the past [Burgard 2005, Wurm 2008, Yamauchi 1998, Gautam 2015]. Frontier based exploration was suggested in [Yamauchi 1997] and has become a de-facto standard for area exploration in multi-robot system studies. A boundary between the known and the unknown region is referred to as a frontier. Robots move to the frontiers and add new information to their maps. In [Yamauchi 1998], the authors extend the basic frontier-based approach to multiple robots, wherein each robot greedily selects the nearest frontier cell to explore. There is a lack of coordination among the robots. As a result, two or more robots that are close to each other sometimes end up choosing the same frontier cell. Some approaches target local dispersion where the robots are dispersed just enough such that their sensory impressions do not overlap, while at the same time remaining in the communication range of at-least one other robot [Mukhija 2010, Rooker 2007]. On the other hand, some works target global dispersion. Either a centralized solution or greater inter-robot communication range [Wurm 2008] is required. In large and complex environments sometimes these approaches result in redundant coverage and take longer time to finish. To corroborate this fact, an excellent comparative study of exploration algorithms has been conducted in [Juliá 2012]. A coordinated multi-robot exploration approach is suggested in [Burgard 2005] where the utility of reaching a given frontier cell and the cost of visiting the same frontier cell are considered simultaneously. Once a frontier cell is assigned to a specific robot the utility of the unexplored region visible from the allotted frontier is reduced, forcing other robots to choose different targets. In [Zlot 2002], a market economy based approach is suggested for the assignment of cells lying in the unknown region to the robots. Robots negotiate the targets that are assigned to them with each other by way of auctioning and bidding. However, [Burgard 2005] is a centralized approach, and in [Zlot 2002], as the size of the environment increases, the length of the message (bid) increases, thus increasing the volume of communication. More recently, instead of allocating the frontier cells, researchers have proposed approaches to allocate an entire segment of the unknown region to each robot [Wurm 2008, Puig 2011, Hungerford 2016]. In [Wurm 2008], a Voronoi graph representing the environment is constructed and certain critical points are obtained at doorways and narrow passages. The graph is then partitioned into different segments, which are assigned to individual robots using the Hungarian method. This approach reduced the overhead incurred

due to repeated allocation of frontier cells. However, if there are very few critical points in the graph, a good segmentation of the map cannot be obtained. Also the approach is centralized and is not robust to failures. In [Puig 2011], K-means clustering is used for partitioning the terrain. Frontier cells in the interior and exterior of the partition allotted to the robot are assigned different costs. This approach demands a large number of computations to repeatedly cluster frontier cells in the unknown region, especially when the size of the environment is large. In a recent work [Hungerford 2016], Voronoi partitions are constructed in a distributed fashion. Each robot respects the territorial boundaries of the Voronoi partitions it falls in, and does not cross the boundaries of its partition. Each robot explores the boundary of its respective regions to determine patches in its region that are unreachable due to obstacles. These patches are then auctioned to other robots. This method guarantees complete, non-overlapping coverage in arbitrarily complex terrains. However, this method does not guarantee even distribution of the workload in the multi-robot system i.e., some robots might explore large regions while some other robots sit idle. More importantly, this method does not handle failure(s) of one or more robots. In [Gautam 2015], instead of allocating large unknown segments to the robots, subsets of tasks comprising of known frontier cells that are clustered using the Geodesic K-means algorithm [Asgharbeygi 2008] are allocated to the robots using the Hungarian method [Kuhn 1955]. In this approach, the dispersion of robots is an emergent behavior i.e., in the beginning of the coverage task when less number of frontier cells are known, the robots are locally dispersed and when the map unfolds and more number of frontier cells become visible the robots start dispersing from each other. A very simple yet intuitive mechanism for terrain coverage is suggested in [Gonzalez 2003]. This approach is based on structured trajectories, such that, simple regions are covered by a single robot using spiral trajectories. As the spiral paths end at the middle of the rectangular region, a backtracking mechanism is introduced to detect and link other regions. It has been argued that structured patterns are more efficient as they sweep/cover long contiguous segments and require the robots to take less number of turns. A multi-robot extension of [Gonzalez 2003] is suggested in [Gerlein 2011]. However, this approach adopts a trivial allocation scheme and results in greater redundancy in coverage. This scheme still does not handle robot failures. In Chapter 4 and 5 of this thesis, we have suggested efficient algorithms for online terrain coverage [Gautam 2015, Gautam 2016b] that are better than some of the representative state of the art approaches. Table 2.2 gives a comparison of the features and capabilities of the different state of the art algorithms.

Table 2.2 A Comparative Analysis of Related Literature. Note O: Online, D: Distributed, C: Complete, and R: Robust (to robot failures)

Approach	O	D	C	R
[Yamauchi 1998]	✓	✓	✓	
[Burgard 2005]	✓		✓	
[Puig 2011]	✓		✓	
[Sheng 2006]	✓	✓		
[Mukhija 2010]	✓		✓	
[Gerlein 2011]	✓		✓	
[Senthilkumar 2012]		✓	✓	✓
[Hungerford 2015]	✓	✓	✓	
[Gautam 2015]	✓		✓	
[Gautam 2016b]	✓	✓	✓	✓

2.4 AREA PARTITIONING / DECOMPOSITION

Multi-robot systems are deployed in an unknown environment for several reasons including mapping and exploration, area coverage, search and rescue, surveillance, patrolling etc. For these tasks the decomposition of the unknown environment for the purpose of robot navigation and path planning is one of the chief characteristics of all the solutions suggested so far. The polygon based decomposition method is shown to be a better mechanism compared to grid based decomposition because the robots quickly disperse and divide the environment into sub-regions which are then processed in a mutually exclusive manner [Wurm 2008].

In [Solanas 2004], the authors have suggested a centralized approach for multi-robot exploration of the unknown environment with known bounds. The environment is decomposed into grid cells using approximate cell decomposition. There are no communication range constraints. The environment is clustered into regions using K-means algorithm. Different clusters are then allocated to the robots for exploration. Once, any robot reaches its cluster centroid, re-clustering is done. This is a computationally expensive approach which results in uneven exploration. In [Wu 2007], for multi-robot exploration, Voronoi based decomposition is used for spatial partitioning of the unknown environment. This reduces the computational cost incurred by using K-means clustering in [Solanas 2004]. The Voronoi partitions are assigned to the robots for exploration. When any one robot

reaches the centroid of its partition, the remaining unknown region is re-partitioned. This is a centralized approach which results in uneven exploration. In [Fu 2009], the robots locally compute the Voronoi diagrams and therefore make independent decisions for processing their assigned regions. Although the approach is distributed it makes no claim for balancing the workload of the robots based on the area each robot has to process. In [Fazli 2010], the authors have suggested a fault-tolerant algorithm for area coverage of the robots. Static guard points are computed in a known map. Using these guards, a graph is constructed which is decomposed into a forest of partial spanning trees (PST) using Multi-prim algorithm. PST's are further processed to make a constrained spanning tour which is then followed by individual robots. This approach also makes no claim about balancing the workload of robots and produces uneven distribution of the tasks. In [Sheng 2006], the authors have suggested a market based approach for area exploration. The robots bid for the exploration task based on expected information gain and distance to a particular location in the environment. The robots avoid areas out of the communication range when the nearness measure as a decision parameter is factored into the bids. This approach allows only local dispersion. Again no claim of load balancing among the robots is visible. Genetic algorithms are used to perform area coverage in [Kapanoglu 2012]. A centralized path planning is adopted in a known map. Again no attempt has been made towards balanced distribution of the workload and the approach is centralized. In [Gautam 2016c], the authors have proposed a distributed algorithm for balanced workload distribution among the individual robots. The workload is defined in terms of the distance travelled by the robot tank team for accomplishing a common objective of destroying the enemy inventory. It is a static multi-robot task allocation problem wherein the location of the tasks and the map of the environment is known upfront.

Based on the above literature survey we can see that the problem of load balancing in task allocation to robots is largely unaddressed. In Chapter 6, a modified approach is proposed for load balanced partitioning of the unknown region.

CHAPTER-3

GEOMETRIC PATTERN FORMATION BY MULTIPLE MOBILE ROBOTS

3.1 INTRODUCTION

It is advantageous to keep multiple robots in a formation when they are dispatched for complex missions like battlefield surveillance, environmental monitoring, etc., as it improves the quality of data collection, task completion time etc. Maintaining a formation requires the robots to execute certain coordination strategy. Multi-robot systems in their organization are expected to be decentralized, such that, individual robots should use only local information to implement their coordination strategy. However, there are situations when they are expected to move with high precision in a predefined strict geometric shape for e.g., parallel and simultaneous transportation of load which requires global knowledge about the work environment and also requires explicit inter robot communication. In the following sub-section our main motivation behind conducting research in geometric pattern formation using multiple mobile robots is discussed.

3.1.1 Research Motivation and Contribution

Theoretical algorithms for geometric pattern formation are proven to be sound and complete under certain assumptions that are way too simplified. For example, the robots are treated as point objects which can sense and move with infinite precision, etc. Such assumptions not uncommon to make when formulating models/solutions for robotic systems. It is understood that these assumptions will not be strictly met in an actual implementation. Nevertheless, it is possible to find approximate solutions for these assumptions. Therefore, such theoretical approaches cannot be compared with other empirical approaches unless someone translates them in an actual implementation. In that sense, the uniform circle formation problem (UCF) with a team of autonomous mobile robots has received considerable attention. The circle is one of the simplest shapes amongst all geometric shapes and therefore it has become a benchmark for such studies. Following are our main contributions described in this chapter:

(a) Conducted a systematic study of various properties and assumptions considered in theoretical studies on geometric pattern formation using multi-robot systems and highlighted various implementation issues.

(b) Presented a model for practical implementation of algorithms for geometric pattern formation after finding approximate solutions of various assumptions.

(c) In our initial attempts towards solving UCF problem with multiple mobile robots, a software framework was designed and implemented to support the implementation of two different algorithms for geometric pattern formation in simulation. The first algorithm is a centralized and greedy algorithm and the second algorithm is weakly centralized and is based on token passing.

(d) A completely decentralized algorithm, referred to as STATE [Gautam 2016a], for solving the UCF problem is proposed. This algorithm does not require assumptions of atomicity, unlimited visibility, global clock etc. The STATE algorithm is shown to perform better than the algorithm suggested in [Défago 2002, Défago 2008], henceforth referred to as DK algorithm. The STATE algorithm is purely distributed in nature and makes use of conflict resolution graphs for multi-robot synchronization. In the proposed algorithm, multi-robot synchronization is achieved by way of message passing in the absence of global clock.

(e) An independent software solution for multi-robot localization has been developed and implemented in a controlled laboratory setting.

(f) A communication sub-system is designed and implemented which can be leveraged for peer to peer communication between multiple robots equipped with Bluetooth technology. The e-puck robots (used in this work) are one of the many instances of the robots equipped with Bluetooth.

The rest of the chapter is organized into seven sections. The problem of UCF is formally stated in Section 3.2. As our initial attempts, a software framework for supporting the implementation of algorithms for geometric pattern formation and two algorithms have been proposed for solving the UCF problem which are discussed in Section 3.3. From the perspective of theoretical computer science, significant properties and assumptions regarding the capabilities of robots, their world models, and the interrelationship between the two, are studied in Section 3.4. Also one of the benchmark algorithms for solving the UCF problem i.e. the DK algorithm is discussed in Section 3.4.1. In Section 3.5, we have proposed a distributed algorithm for solving the UCF problem i.e. the STATE algorithm. The STATE algorithm employs the distributed solution of the dining philosopher problem for achieving synchronization in multi-robot system. The design of experimental test-bed/setup is

discussed in Section 3.6. A statistical comparison of the implementations of the DK algorithm and the STATE algorithm is presented in Section 3.7. We have conducted Wilcoxon test to substantiate our results and show that the STATE algorithm performs better than the DK algorithm. The chapter is summarized in Section 3.8.

3.2 PROBLEM STATEMENT

The problem of uniform circular formation with a team of autonomous mobile robots is expressed as follows:

Given a group of N autonomous mobile robots arbitrarily scattered on a two dimensional plane, in finite time should develop consensus to arrange themselves on the periphery of a non-degenerated circle, and further should position themselves at regular intervals, such that, the angular distance θ between any two direct neighbors is $2\pi / N$.

3.3 INITIAL ATTEMPTS

The main objective of this chapter is to suggest a completely decentralized yet efficient algorithm for circle formation and practically validate the efficacy of the same. In this section, we present our initial attempts towards the stated objective.

3.3.1 Software Framework

In this section, we have proposed a software framework [Gautam 2012] for supporting the implementation of the two different algorithms for solving the UCF problem. The high level design of the framework is presented below:

High-Level Design (HLD)

The software framework is presented in terms of UML class diagrams in Figure 3.1 and Figure 3.2. The suggested framework makes use of two well-known software design patterns i.e. the observer and decorator pattern [Gamma 1995]. The robots are arbitrarily placed in the environment. In the beginning none of the robots can be discriminated as leader or follower. A robot can function either as a leader or as a follower. Two different decorators i.e. robot leader and robot follower are created. The leader robot is determined using a leader election algorithm like the one suggested in [Dieudonné 2007]. We decorate the elected leader robot with the responsibilities of a leader. All other robots are decorated with follower responsibilities. Two concrete decorator classes namely RobotLeader and RobotFollower are used for this purpose and are shown in the class diagram of Figure 3.1. The robot leader registers all the robot followers, computes and notifies appropriate positions for all of them

on the circle circumference. The observer design pattern provides the mechanism for information exchange between leader and followers. Figure 3.2 represents RobotLeader as a Subject and RobotFollower as an Observer. RobotFollowers use RobotSubject interface to register as Observers and also to remove themselves from being an Observer. This interface has just one method that gets called when the Subject's (i.e. RobotLeader's) states change. RobotLeader is a concrete subject which implements RobotSubject interface and has implementations to register, remove and a method to notify observers. RobotFollower is a concrete observer as it implements the RobotObserver interface. Each RobotFollower registers itself with RobotLeader to receive updates. With this HLD in place we have suggested two algorithms in the following section for solving the UCF problem. The two algorithms are implemented in simulation.

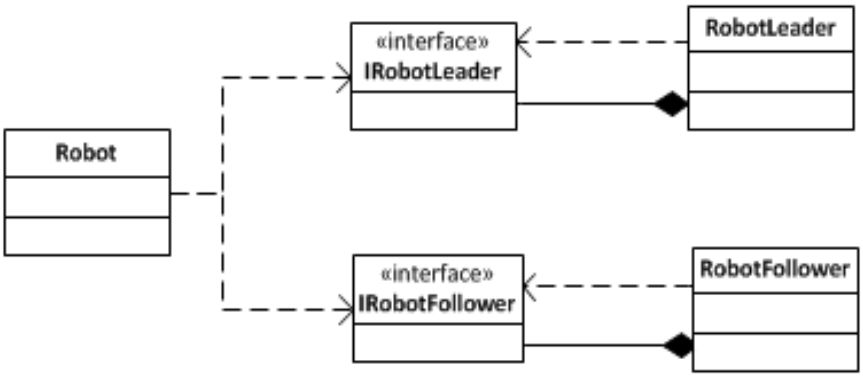


Figure 3.1 Decorate Robot as Leader or Follower

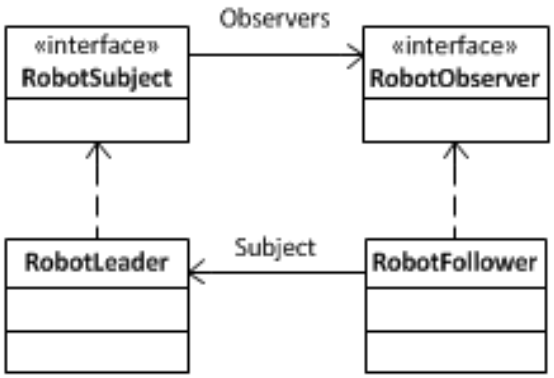


Figure 3.2 RobotLeader as Subject and RobotFollower as Observer

3.3.2 A Centralized Algorithm for UCF

A centralized algorithm for solving the problem of UCF is proposed in this thesis [Gautam 2012]. The proposed algorithm starts with the assumption that the leader robot or *RobotLeader* is already elected and decorated with the role of a leader and all other robots

i.e. the follower robots or *RobotFollower* are decorated with the role of the follower. The algorithm executed by the leader robot for solving the UCF problem is presented below:

1. *Calculate Smallest Enclosing Circle (SEC)*: Assume a set *P* of *n* points representing the position of all *RobotFollowers*, $P = \{p_1, p_2, p_3, \ldots, p_n\}$ in the Euclidian plane $\Re^2$. The smallest enclosing circle [Skyum 1991] of *P*, SEC(P), is the circle with minimal radius enclosing all points in *P* and is shown in Figure 3.3. It is also well known that SEC(P) = SEC(H), where *H* is a proper subset of *P*, consisting of extreme points on the convex hull of *P*.
2. *Calculate Uniform Positions on the Circumference of SEC*: The *RobotLeader* is positioned at the center of SEC, shown as star in Figure 3.4. It calculates the uniform positions for the *RobotFollowers* on the circumference of SEC.
3. In Figure 3.4 the star and triangle robots are active, such that, the one that is on the center (star) is *RobotLeader* and the other one (triangle) is one of those *RobotFollowers* which was already on the circumference and is randomly picked up as a reference point. It is named *FirstFollower* and is positioned (will not move). Now the *RobotLeader* finds the coordinates of *n-1* points for *n-1* remaining robots beginning from *FirstFollower*, such that, all points are separated from each other by an angular distance of θ degrees:

   $\theta = 2\pi/n\text{-}1$, where *n* is the number of *RobotFollowers*
4. Now we know *x* and *y* coordinates of *RobotLeader*, *FirstFollower*, and an angle θ. We want to determine the *x* and *y* coordinates of cross points shown in Figure 3.4 on the circumference of SEC.

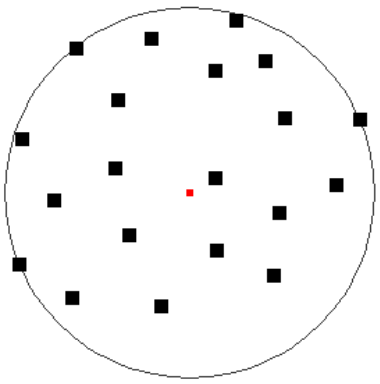


Figure 3.3 Smallest Enclosing Circle of a Set of Points Representing the Position of RobotFollowers

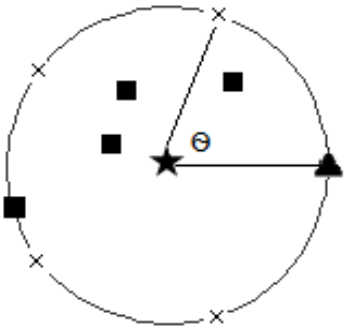


Figure 3.4 SEC with two Robots on circumference and three Robots inside it

5. We calculate two points x and y as follows:

$$x = FirstFollower.x - RobotLeader.x$$

$$y = FirstFollower.y - RobotLeader.y$$

To determine *x* and *y* coordinates of cross points we run a loop from *i = 1 to n-1* and perform the following calculations:

$$a_i.x = RobotLeader.x + x*cos(i*\theta) - y*sin(i*\theta)$$

$$a_i.y = RobotLeader.y + y*cos(i*\theta) - x*sin(i*\theta)$$

This way we determine all uniform positions where *RobotFollowers* are finally positioned. The running time of this algorithm is *O(n-1)*.

6. *Finding right position for RobotFollowers*: *RobotLeader* maintains a list of uniform positions. Leaving the *FirstFollower* and its position, *RobotLeader* runs an optimal allocation algorithm for finding the position of each *RobotFollower* on the basis of distance cost. All *RobotFollowers* are notified their assigned positions on the circumference of the SEC. *RobotFollowers* move to their respective positions. As a result, circle formation by arbitrarily scattered multiple mobile robots in 2D plane is achieved.

3.3.3 A Weakly Centralized Token Passing Approach for UCF

A weakly centralized approach for positioning multiple mobile robots in a circular formation based on token passing [Gautam 2013a] is presented in this section. This algorithm is an extension of the previously proposed centralized algorithm. It is also a leader-follower approach wherein it is the leader robot which computes the uniformly spread out positions on the circle circumference for all the follower robots. The problem of circle formation is divided into two sub-problems (a) leader selection and (b) finding positions for the follower

robots from the set of uniform positions computed by the leader robot. Both these problems are solved by way of token passing so as to reduce the burden of position assignment to all the followers from the leader robot. The introduction of token passing makes it a weakly centralized algorithm. This algorithm starts with the assumption that the leader robot is already elected and it has computed all the positions for the follower robots on the circumference of the SEC. These positions are notified to the follower robots. Only those follower robots that are outside the circle with the radius $\frac{SEC_{radius}}{2}$ greedily select their own position on the basis of their distance from that position and broadcast this information. It is possible that more than one robot may select the same position to move to. All such robots form a virtual token ring amongst themselves based on their IDs and eventually bid for that position by way of token passing. The bid is nothing but the distance of the robot from the position under consideration. The position is allocated to the robot with minimum bid. The process repeats until there are claimants for some position. Otherwise, if some positions are left unoccupied and some robots are not assigned a position. The leader robot assigns all the unoccupied positions to all the unallocated robots. This algorithm relieves the leader robot of the burden of position assignment to all the robots.

The two algorithms we have discussed in this section are centralized. Moreover, the second algorithm results in a sub-optimal assignment. Also they are unrelated to the theoretical studies and algorithms presented in the literature on UCF. Therefore, the discussion on these two algorithms is limited to this section only. In the next section, we have systematically studied various properties and assumptions considered in theoretical studies on geometric pattern formation using multi-robot systems and highlighted various implementation issues. Followed by that, a discussion on one of the representative state of the art algorithms for solving the UCF problem i.e. the DK algorithm is presented.

3.4 Analysis of Significant Properties and Assumptions

Theoretical research on geometric pattern formation using multiple mobile robots is based on the assumption that the robots are simple and have limited capabilities. Thus, the perceptual capabilities of the robots are abstracted. If not impossible, it is difficult to translate most of the assumptions into reality without compromising the model itself. It is important to understand how these assumptions can be approximated. Following is an analytical discussion of several assumptions considered in [Suzuki 1999, Prencipe 2001, Défago 2002, Souissi 2004, Chatzigiannakis 2004, Flocchini 2005, Défago 2008, Krick

2009, Flocchini 2008, Dieudonné 2008, Dieudonné 2009, Elor 2011]:

(a) *World model*: The world model of the robot is an unbounded two dimensional plane without any noise. In actual implementation the noise in the environment greatly influences robot operations and should be addressed.

(b) *Robot's dimension*: The robots are considered as a point on a plane (dimension less). In the real world, robots have some finite dimension which should be considered when designing algorithm(s) for multi-robot systems.

(c) *Mobility*: It is considered that the robots move freely with infinite precision in the world. Real robots, however, are required to continuously localize themselves, avoid obstacles and plan their paths to their destination.

(d) *Visibility*: It is assumed that the robots are equipped with sensors which instantaneously return the location of all other robots with infinite precision. Such an advanced sensing system for simple mobile robots is a very strong assumption. It completely eliminates the need for localization. In multi-robot systems the robots are typically equipped with sensors with limited capabilities and therefore motion capture systems are heavily used in the indoor environment. Localization is achieved with the help of an assembly of several high speed cameras which can track the reflective markers on the robots at the rate greater than 300 fps. One can expect to achieve precision of less than 1 mm by using this system. In outdoor scenario, the robots that are used are neither low cost nor they are simple in design. For the purpose of simultaneous localization and mapping (SLAM), robot's odometry with laser scanners [Howard 2006] and vision based methods are frequently used [Meltzer 2004].

(e) *Anonymity*: It is assumed that the robots are identical and they cannot be differentiated from each other. For geometric pattern formation the robots are required to maintain a certain distance and orientation with their peers. It can only be achieved with accurate localization of the peers. This fact is also substantiated in several experimental research works [Fredslund 2002, Mead 2009]. In practice mobile robots continuously localize themselves with respect to the features present in the environment. The robots should be able to sense other robots (stationary or moving) and non-robot entities in the world. Therefore, anonymity contradicts the need for localization of the peer robots. Incidentally the unlimited visibility

assumption supports the assumption of anonymity.

(f) *Communication*: The robots communicate and decide their own action only by implicitly observing other robots. The assumption of anonymity enforces implicit communication because robots do not know each other. Therefore, implicit communication has a profound reliance on the assumption of unlimited visibility and is difficult to realize with weak robots. A real multi-robot system, in the presence of sensor inaccuracies, noise in the environment, and without explicit communication, will end up wasting a lot of energy in cancelling disturbances. It might actually fail to complete its mission of uniform circle formation. In the work presented in this thesis, we have dropped the assumptions of anonymity and implicit communication. An algorithm has been designed, with explicit inter-robot communication model.

(g) *Autonomy*: It is assumed that the robots are autonomous. Autonomous robots resist any type of external intervention in their operations. The success of the mission depends on the coherent behavior of individual robots, i.e., how well they coordinate and adapt to the movements of other robots. A robot involved in the task of geometric pattern formation requires precise knowledge of the initial placement of its peers. Further, the robots are arbitrarily deployed in their environment. They are expected to adapt to this initial placement and start their mission as soon as they are activated. The system is said to be self-stabilizing, if it makes continuous progress and converges to its final state of uniform circular formation. It is not trivial to coordinate the actions of mobile robots in a multi-robot system, especially when there is no explicit communication and coordination.

(h) *Coordinate system*: No global reference frame is assumed. Each robot operates in its own local coordinate system. This essentially means that all the percepts are received in robot-centric coordinates. Therefore, based on only local computations the robots decide their future course of action. Again the robots need precise information (location and heading) about themselves and their peers.

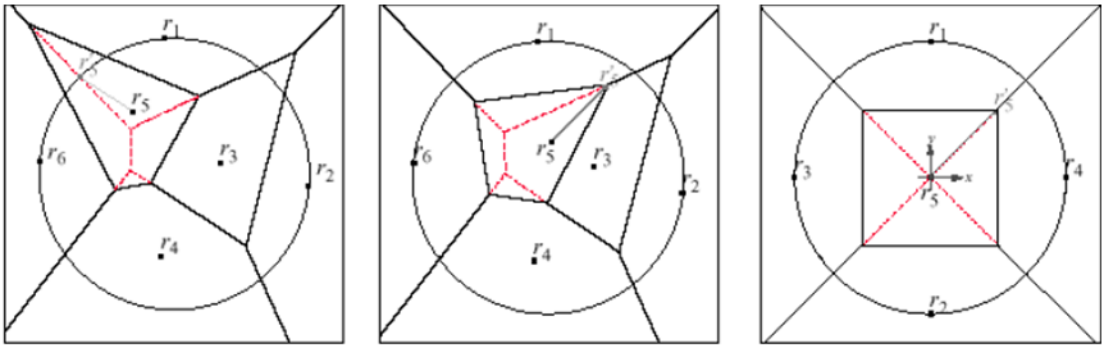
(i) *Agreements*: Partial agreement on the local coordinate system of robots is assumed, i.e., the direction and orientation of one of the axis, say the X axis is known to the multi-robot system. In [Prencipe 2002], the authors have proved that, with common knowledge of the direction and orientation of the two axes, multi-robot system can form an arbitrary given pattern. Also, if the robots only have a partial agreement,

then, there is no deterministic algorithm that permits an even number of robots to form an asymmetric pattern. They can only form symmetric patterns. The circle is a symmetric pattern and can be formed irrespective of the number of robots. In real multi-robot implementations, to be able to approximately translate this assumption, one needs highly precise heading sensors.

(j) *Synchronization*: The time is represented as an infinite sequence of discrete time instances $\{t_1, t_2, t_3,\ldots,t_n\}$. It is assumed that a global clock tick is reaching the robots, such that, at any time instance t_i only a set of robots become active. Additionally, it is also assumed that the activation cycle LOOK-COMPUTE-MOVE-WAIT is instantaneous and the time to execute one cycle is negligible. This is a very strong assumption because in practical scenario the time to complete any single activation is going to be arbitrary and finite and cannot be determined a priori. Thus, it is very likely that between any two different time instances t_i and t_j some robots get activated. This violates the basic assumption of atomicity considered in SYm and brings the system in an inconsistent state.

3.4.1 The Défago and Konagaya's (DK) algorithm

The Défago and Konagaya's (DK) algorithm [Défago 2002, Défago 2008] is briefly described in this section. This algorithm relies on two assumptions: (a) the work environment of all the robots is invariant irrespective of the fact that the robots are ego-centric and (b) the smallest enclosing circle is exclusive and invariant. The DK algorithm for solving UCF has two parts. In the first part, the robots are assumed to be arbitrarily positioned in the environment with distinct positions; the algorithm $\emptyset_{circle}$ is suggested that brings the system into a configuration wherein each robot is sitting on the circumference of a smallest enclosing circle (SEC). Once all the robots reach the circumference of the circle, the second part an algorithm $\emptyset_{uniform}$ schedules and makes the robots move towards achieving a uniform circular formation. Combining the two algorithms $\emptyset_{circle}$ and $\emptyset_{uniform}$ solves the problem of UCF.



(a) The SEC is Reachable **(b) The SEC is Unreachable** **(c) The Symmetry is Broken**

Figure 3.5 Pictorial representation of the algorithm $\emptyset_{circle}$ independently executed by each robot r_i

The basic idea behind the algorithm for circle formation i.e. $\emptyset_{circle}$ is very simple: The robots that are already positioned on the circumference of the SEC are not allowed to move and those that are positioned in the interior of the SEC progress towards the circumference. After computing the Voronoi diagram, the robots that are positioned in the interior of the SEC may find themselves in three different situations:

- In the first situation, the Voronoi partition of all the robots intersects the SEC, as shown in Figure 3.5(a). Therefore, the robots select the point of intersection of its Voronoi cell with the SEC and reach there.
- In the second situation, the Voronoi partition of some robots does not intersect with the SEC. Such robots select a point inside their Voronoi partition which is closest to the circumference of the SEC, as shown in Figure 3.5(b).
- In the third situation, for some robot(s) several points exist on the circumference of SEC, due to symmetry, as shown in Figure 3.5(c). In this case any one point is randomly selected.

Once the circle is formed (i.e. all robots are positioned at the circumference of the SEC), each robot starts executing the uniform circle formation algorithm i.e. $\emptyset_{uniform}$. The algorithm is described below:

- Whenever some robot r_i gets activated it travels halfway distance towards the mid-point between the two of its immediate neighbors. This process continues until the system converge towards the UCF.

- A typical situation exists in this algorithm wherein the system starts oscillating endlessly between the two configurations if the robots are perfectly synchronized. Only to overcome this situation the robots are allowed to move half-way towards the mid-point between the two neighbors. The convergence of this algorithm is proved in [Souissi 2004].

In the following section we present a detailed discussion on the proposed algorithm i.e. the STATE algorithm which is empirically shown to perform better than the DK algorithm.

3.5 PROPOSED APPROACH TO SOLVE UCF PROBLEM

To make multiple mobile robots fall in a uniform circular formation we present a completely distributed algorithm in this section. Before we discuss the details of the proposed algorithm, a new system model is presented in the following sub-section.

3.5.1 System Model

The system model proposed in this section is highly suitable for empirical analysis of algorithms for geometric pattern formation of multiple mobile robots. It consists of a set of real mobile robots, $R = \{r_1, r_2, ..., r_n\}$. We have used a team of five e-puck robots [Mondada 2009] which are small differential wheeled mobile robots for experiments. The robot's working environment is a two dimensional plane ($\Re^2$). Each robot is powered by a dsPIC processor. The robots do not take into account their previous states and actions and are therefore *oblivious*. They can move freely on the plane and are *mobile*. Each robot operates in its own *local coordinate system* which includes an *origin* (which is the current position of the robot), *direction and orientation of the positive X axis*. The robots have no agreement with each other on the unit of length, origin of the circle, the orientation of the coordinate axes, and speed. All robots have a *unique identity* (ID). A special marker is attached on the top of the robots for identification. Localization in multi-robot systems is a different research problem and we are not explicitly targeting the same in this work. Instead, we have used an overhead camera connected to a computer that reads the markers attached on the top of the robots and stores the *current system configuration*, i.e., the ID, position and orientation of all the robots (henceforth referred to as *CSC*). This system is referred to as *localization sub-system* and is discussed in Section 3.6.1. Although we are not carrying out localization on-board the robot, our system should not be considered as a centralized system. Localization of mobile robots is an abstraction here. All the robots are *autonomous* and *independently* execute the proposed algorithm for uniform circle formation, perform obstacle avoidance

and path planning. Our model is a semi-synchronous model where robots synchronize with their peers by way of message passing using wireless communication.

3.5.2 Activation Cycle of Robots

Each robot in this model repeatedly goes through a sequence of four states (also known as activation cycle) as shown in Figure 3.6. Each state of the activation cycle is explained below:

(a) *Idle or Wait* (W): Initially, all the robots are in the waiting state, but they cannot remain permanently in this state.

(b) *Sense or Look* (L): When the robot becomes active it pulls the *CSC* from the localization sub-system and transforms the information in its own robot-centric coordinates.

(c) *Compute* (C): The robot carries out local computations according to a deterministic algorithm. This algorithm is same for all the robots. The algorithm returns a destination point for the robot to move to.

(d) *Move* (M): The robot moves towards its computed destination. The distance travelled by a robot in the single activation cycle is finite.

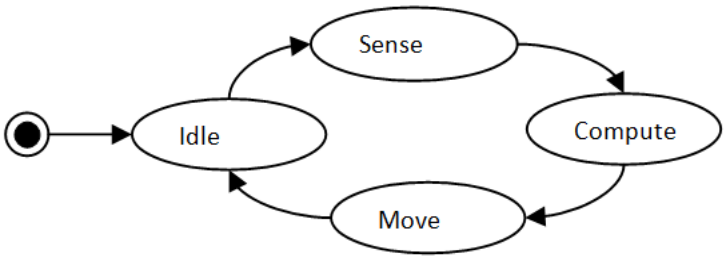


Figure 3.6 Activation Cycle of a Robot

3.5.3 Formulating the Activation Cycle of Robots as a Critical Section Problem

In the proposed model, the activation cycle of the robots is not instantaneous i.e., for the robots to execute their activation cycle it will take some arbitrary amount of time. The amount of time a robot spends in any of the four states of its activation cycle is not known a priori. For a physical mobile robot, sensing, computing and moving to a destination is going to take some time which is bounded but arbitrary in magnitude. This is a practical condition which makes the uniform circle formation problem more complex.

If the robots are allowed to operate with complete autonomy it will result in race conditions[1]. This will bring the system configuration in an inconsistent state. Here the robot's activation cycle is required to be governed by some protocol, which does not allow the interleaving of the states. In other words, the activation cycle of a robot should be executed atomically.

This problem can be formulated as a critical section problem where multiple processes compete for some shared system resources. We present an analogy that will help us develop a better understanding of the situation. Multiple mobile robots operate in the same environment and therefore it is analogous to a shared resource. The very first state in the activation cycle of the robot is the idle/wait state which is analogous to the entry section of the operating system process. In this state the robots get activated according to the activation policy employed (discussed in the subsequent section). Only active robots attempt to get into the critical section. The other three states, sense/look, compute, and move, fall in the critical section. Those robots which are in the critical section switch to the look state and the other robots remain in the wait state. In the sense/look state the robots pull the *CSC*, which is equivalent to a read operation on the environment. All robots in the critical section compute their next target position based on *CSC* and switch to move state. The move state changes the state of the system configuration and therefore it can be considered as a write operation on the environment. After reaching their destination and before switching back to the wait state, all the robots flush the *CSC* sensed earlier, which is analogous to the exit section of a process. There are no instructions in the remainder section. Figure 3.7 depicts the analogy presented above. The view presented in this figure is hypothetical in the sense that operating system processes which typically run on uni-processor or multi-processor machines make use of shared memory and solutions like semaphores, monitors, and test-and-set instructions are used for inter-process communication and synchronization.

[1] This work doesn't make any assumptions on time, i.e. the robots do not share a global clock nor do they try to synchronize their local clocks. The time duration for which a robot remains in a particular state (as described in the activation cycle of the robot) is unpredictable but finite. As a result of this unpredictability a complex situation arises. The robot before calculating its next target position senses the current positions of its neighbors. It is very much possible that the movement the robot performs is not coherent with the current context (the positions of the neighbors it has observed during its sense state and the position of the neighbors at the time when it executes the move state can differ) because different robots might be in different states of their activation cycle at a given time instance.

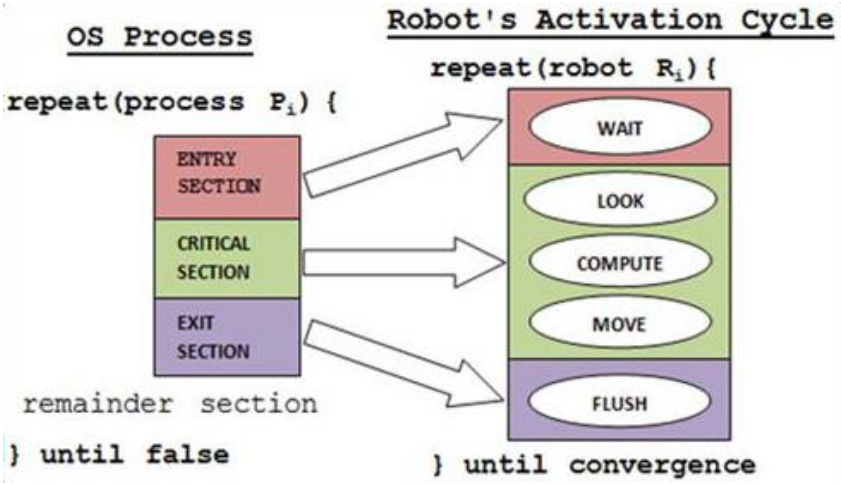


Figure 3.7 Structure of Robot's Activation Cycle

In multi-robot systems, each robot is a computational entity and does not support sharing of files or memory with other robots. Basically, a multi-robot system is a distributed computing system where each robot carries out local computations, share data and critical resources by sending and receiving messages to and from other robots. No locking primitives that are defined for operating system processes can be directly used in a distributed computing system. Thus, it is a problem of distributed mutual exclusion. In the next sub-section, we have explained how distributed multi-robot synchronization is achieved with the activation policy/schedule suggested for DK and the proposed STATE algorithm.

3.5.4. Distributed Multi-Robot Synchronization and Activation Policy

Probabilistic scheduling is employed in DK algorithm [Souissi 2004] and it is shown that more the robots are involved with each other (act tightly in synchronization with each other) faster they are able to accomplish their mission of uniform circle formation. This requires clock based synchronization of multi-robot systems. A global clock is assumed in DK algorithm for the same reason. On the other hand, the proposed approach employs the solution suggested in [Garg 2004] for distributed mutual exclusion. No clock is required as mutual exclusion is implemented by event ordering. The algorithm enforces a particular order on robot's activation and hence we call it order preserving schedule. Following is a more detailed discussion of the above two scheduling policies:

(a) *Probabilistic schedule*: We need to revisit the question as to how a multi-robot system exhibits semi-synchronous behavior in Suzuki-Yamashita model. In DK algorithm a global clock tick is assumed to be reaching all the robots. A robot gets activated after receiving a clock tick and finishes executing its activation cycle instantaneously in

negligible time. This entails that all active robots atomically complete their activation cycle at the same time in equal duration. As a result, the robots appear to be in semi-synchronous mode of execution. A special case is reported when the robots execute in complete synchronization, i.e., all robots are activated every clock tick. The activation of a robot is assumed to be probabilistic, such that, probability (P_r) = 1 indicates fully synchronous behavior. Lower the probability of activation, lesser the number of robots active during each time instance. In [Souissi 2004], a probability (P_r) = 0.5 allows a sufficient number of robots to remain active in each round and as a result the system exhibits a semi-synchronous behavior. No particular order is prescribed for robot's activation, i.e., at every time instance, a robot which is sitting on the circumference of the circle may become active while its neighbors (clockwise and anti-clockwise) are also active. Also one robot may get activated more than once in consecutive clock ticks while its neighbors (clockwise and anti-clockwise) may remain silent. Such arbitrary activations do not add value to the system as far as convergence towards a uniform circular formation is concerned. Instead, it results in increasing the number of activation steps and wastes robot's resources. In our implementation of the DK algorithm on a physical robot test-bed, we have ensured that a new clock tick is sent to the robots only when all the robots which got activated in previous clock tick finish executing their activation cycles. This preserves the semi synchronous behavior of the system.

(b) *Order preserving schedule*: Each robot gets activated in a well-defined order, i.e. if one robot is active its neighbors (clockwise and anti-clockwise) stay silent. This is a problem of *distributed mutual exclusion*. The semaphore and shared variable solution is not usable on distributed systems like ours where each robot is a separate processing unit and do not support sharing of physical memory with other robots. The synchronization protocol used by the robots is a distributed solution of the classical Dining Philosopher problem which is solved without central coordination with message passing and is suggested in [Gautam 2013b, Gautam 2014].

The philosophers have three states, *THINKING*, *EATING*, and *HUNGRY*. Whenever the philosophers are *HUNGRY* they require shared resources, the forks, to eat. Analogous to this the robots (can be thought of as philosophers) also have three states: *IDLE*, *ACTIVE*, and *EXECUTE*. Whenever the robots are *ACTIVE* they wish to execute the steps described in their activation cycle, and for that they require shared resources (forks). Two requirements suggested for the use of shared resources are (i) *mutual exclusion* – shared resources cannot

be shared held by more than one robot at a time and (ii) *freedom from starvation* – every robot should be able to execute infinitely often. The distributed solution to the synchronization problem makes use of an undirected graph called conflict graph, in which each node represents a robot. An edge between two robots R_i and R_j represents that one or more resources are shared between them. It is required that we provide directions to all edges, such that, if the edge between R_i to R_j points from R_i to R_j (denoted as $R_i \rightarrow R_j$) then R_i has a precedence over R_j. If the robot R_i has precedence over robot R_j it will associate a fork with the edge *(i, j)*. The algorithm complies with two rules (a) *Execution rule* – A robot can execute if it has all the forks for the edges incident to it and (b) *Edge reversal* – A robot reverses the orientation of all the outgoing edges to incoming edges. It is important to distinguish between two scenarios, first when the robot has the forks but has not used them, and second, when the robot has the forks and has used them for execution. A Boolean variable "*dirty*" is attached with each fork, so that when a robot has executed with these forks they become dirty. Before a fork is sent to the neighbors they are cleaned. The forks are initialized and placed in such a manner that the conflict resolution graph remains acyclic. It is to be observed that acyclic conflict resolution graph is invariant and the algorithm ensures the same, by cleaning the forks before they are sent. The only action that changes the graph is when the robot executes its activation cycle thereby reversing all the edges incident to it. This edge reversal will never let cycles to be created in the conflict resolution graph and hence no robot ever does famish, which proves the fairness property.

This solution satisfies all the three requirements which a correct solution to any critical section problem must satisfy i.e. mutual exclusion, progress, and bounded waiting, in a distributed manner. In STATE algorithm, we have applied the same activation policy. This policy ensures fairness and all robots get equal opportunity to execute their activation cycles periodically. As a result, the numbers of activation cycles reduce and the overall performance of the multi-robot system improves.

3.5.5. The STATE Algorithm

To make multiple mobile robots fall in a uniform circular formation we present a decentralized algorithm i.e. the STATE algorithm [Gautam 2016]. This algorithm is illustrated with the help of a flowchart shown in Figure 3.8. Also each step of the flowchart demands separate description which is as follows:

Step-1, Sense the position of other robots – Each robot pulls the *CSC* and therefore acquires the knowledge of the *x* and *y* positions of all other robots in its own local coordinate system which is maintained as a set *P* of *n* points, $P = \{p_1, p_2, p_3 \ldots, p_n\}$ in the Euclidian plane $\Re^2$.

Step-2, Calculate the Smallest Enclosing Circle (SEC) – Having the knowledge of the positions of all other robots each robot locally executes the algorithm for finding the smallest enclosing circle which is described in [Skyum 1991]. *SEC(P),* is the circle with minimal radius enclosing all points in *P*. It is also well known that *SEC(P) = SEC(H),* where H is a proper subset of P, consisting of extreme points on the convex hull of *P*. Doing this will implicitly develop a common consensus amongst all the robots to agree upon the definition of *SEC(P),* such that, the coordinates of the center of the *SEC* locally calculated by each robot will coincide with equal radius. The *SEC* is invariant. Three restrictions on the movement of the robots are imposed so as to preserve the invariance of the *SEC* i.e. to prevent the robots from making the movements that may lead to breaking of the *SEC*. The first restriction does not allow the robots to travel outside the boundary of the *SEC*, the second restriction requires that all robots located on the circumference of the *SEC* remains there and the third restriction demands that the robots located on the circumference do not move unless there are at least three such robots with distinct positions. If the *SEC* is defined by only two points, these points define the diameter of the *SEC*. Thus if one of them moves, the circle is broken. Robots are explicitly programmed to comply with these restrictions.

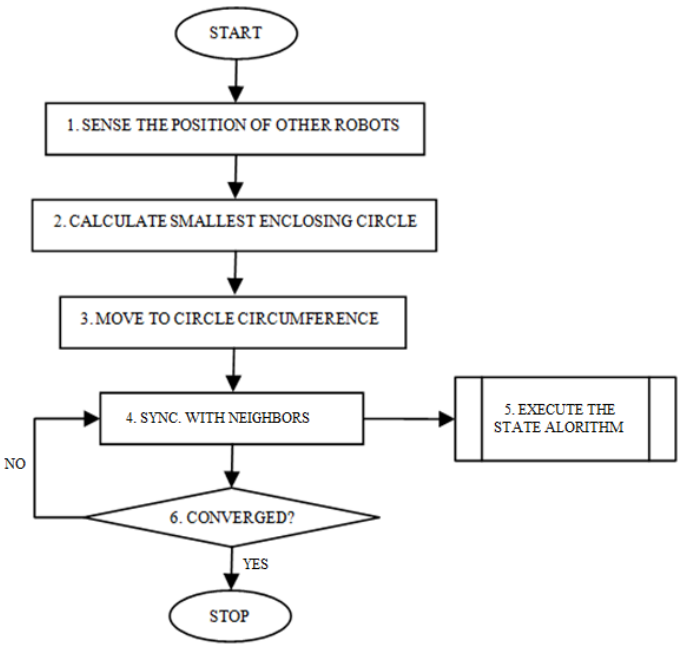


Figure 3.8 Flowchart of STATE algorithm independently executed by each robot

Step-3, Each Robot Moves to the Circumference of the SEC – Using the circle line intersection formulation proposed in [Rhoad 1984] the robot determines the position on the *SEC* circumference which is at the shortest distance and moves to that position. Figure 3.9 illustrates this scenario. The *SEC* is defined by the coordinates of only two robots *R1* which is located at *(x1, y1)* and *R2* which is located at *(r2, y2)*. These two coordinates define the diameter of the *SEC*. Robot *R3* is located at the coordinate *(x3, y3)* which is inside the *SEC*. This robot is required to move on to the circumference of the *SEC*. Considering the coordinates *(c0, c1)* of the origin *C* of the *SEC* and its own coordinates *(x3, y3)*, Robot *R3* produces a secant which intersects the *SEC* at two points *P0* and *P1*. It applies the following formulation for calculating the *X* and *Y* coordinates of the points *P0* and *P1*.

Defining,

$$\mathrm{dx} = \mathrm{x3} - \mathrm{x0} \tag{3.1}$$

$$\mathrm{dy} = \mathrm{y3} - \mathrm{y0} \tag{3.2}$$

$$\mathrm{dr} = \sqrt{\mathrm{dx}^2 + \mathrm{dy}^2} \tag{3.3}$$

$$\mathrm{D} = \begin{vmatrix} \mathrm{x0} & \mathrm{x3} \\ \mathrm{y0} & \mathrm{y3} \end{vmatrix} = \mathrm{x0y3} - \mathrm{x3y0} \tag{3.4}$$

Gives two points of intersection *P0* and *P1* as

$$\mathrm{x} = \frac{\mathrm{Ddy} \pm \mathrm{sgn(dy)dx}\sqrt{\mathrm{r}^2\mathrm{dr}^2 - \mathrm{D}}}{\mathrm{dr}^2} \tag{3.5}$$

$$\mathrm{y} = \frac{-\mathrm{Ddy} \pm |\mathrm{dy}|\sqrt{\mathrm{r}^2\mathrm{dr}^2 - \mathrm{D}}}{\mathrm{dr}^2} \tag{3.6}$$

Where the function *sgn*(x) is defined as

$$sgn(\mathrm{x}) \equiv \begin{cases} -1 \text{ for } \mathrm{x} < 0 \\ 1 \text{ otherwise} \end{cases} \tag{3.7}$$

Robot *R3* calculates the Euclidean distance from both the points of intersection *P0* and *P1* and moves to the one which is at the shortest distance, *P0* in this case. When all the robots are on the circumference of the *SEC* they enter their activation cycle and make a transition to the IDLE state of their activation cycle.

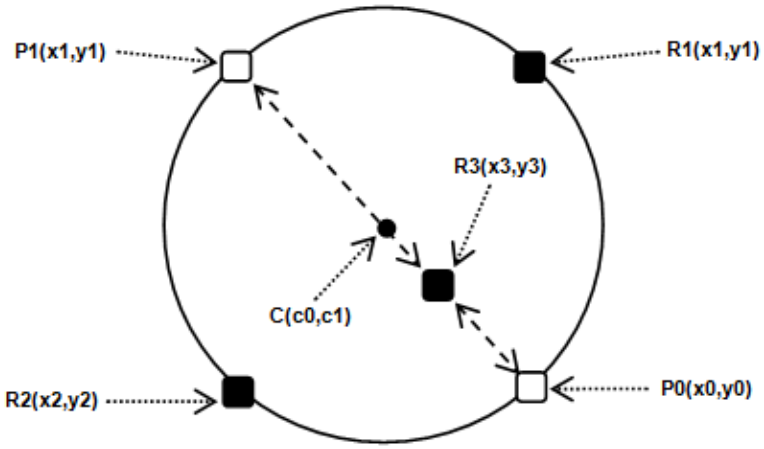


Figure 3.9 Robot trying to determine and move to the position on the circumference of the SEC which is at the shortest distance

Step-4, Run Activation Cycle – All robots infinitely and randomly become active within a bounded time which is proportionate to the number of robots. When a robot becomes active it goes to the *SENSE* state and determines its direct neighbors (clockwise and anticlockwise). At this point before making a transition to the *COMPUTE* state, the robots synchronize (*Step-5*) with their direct neighbors. Synchronization is defined as a sub-process of the activation cycle and is already described in Section 3.5.4. If a robot wins the chance it transits to the *COMPUTE* state and computes the next target position by executing the STATE algorithm as described below:

Algorithm 3.1: STATE Algorithm for Uniform Circle Formation

Notations used:

(a) $\Theta=2\Pi/N$ (N=number of robots) is the *characteristic angle*.

(b) $R = \{r_1, r_2,...,r_N\}$ is the set of N robots where *"r_i"* denotes the current position and the unique ID of the robots on a two dimensional plane.

(c) Given two points *"a"* and *"b"* on a two dimensional plane, *[a, b]* denotes the line segment between *"a"* and *"b"*. $|a, b|$ denotes the magnitude of the line segment *[a, b]*.

(d) Given two line segments *[c, a]* and *[c, b]*, $\angle([c, a], [c, b])$ denotes the angle centered at *"c"* between the line segments *[c, a]* and *[c, b]*. $|\angle([c, a], [c, b])|$ denotes the magnitude of the angle.

(e) The smallest enclosing circle of all robots is denoted by *"C"*, where the center of the circle is denoted by C_{center} and radius by *rad*. We say robot *"r_i"* is on *C* if $|r_i, C_{center}| = rad$.

(f) Clockwise neighbor of robot *"r_i"* is denoted by *next(r_i)* and the counter clockwise neighbor of robot *"r_i"* is denoted by *prev(r_i)*.

(g) $CCW_{position}(x)$, where *"x"* is a point on the circumference, denotes the point such that $|\angle([C_{center}, x], [C_{center}, CCW_{position}(x)])| = \theta$ and $CCW_{position}(x)$ is the counter-clockwise neighbor of *"x"*.

(h) $CW_{position}(x)$, where *"x"* is a point on the circumference, denotes the point such that $|\angle([C_{center}, x], [C_{center}, CW_{position}(x)])| = \theta$ and $CW_{position}(x)$ is the clockwise neighbor of *"x"*.

(i) Given two points *"a"* and *"b"* on the circumference of smallest enclosing circle *"C"*, *mid(a, b)* denotes the mid-point between *"a"* and *"b"* in polar coordinates.

Preliminaries: Each robot can be in any of the three states *{2-Stable, 1-Stable, No-Stable}*. A *2-Stable* state is achieved when the robot is at a polar distance which is equal to angle θ with both of its direct neighbors i.e., clockwise and anti-clockwise. A *1-Stable* state is when the robot is at a polar distance which is equal to angle θ with only one of its direct neighbors either clockwise or anti-clockwise. A *No-Stable* state is when the robot is neither *2-Stable* nor *1-Stable*.

Pre-conditions: $\forall r_i \in R, |r_i, Ccenter| = rad$

Description of Function $\phi_{STATE}(R, r_i)$: Each robot executes the STATE algorithm independently until convergence. The convergence condition is evaluated in a loop, that the robot r_i is subtending the characteristic angle on its direct neighbors (Line-1). Until the convergence condition is satisfied the robot moves to the position returned by $\phi_{main}(r_i)$ (Line-2).

Function $\phi_{STATE}(R, r_i)$: Each robot runs this function in order to calculate its next position until convergence.

1. **while**(!($|\angle([C_{center}, r_i], [C_{center}, next(r_i)])| == |\angle([C_{center}, r_i], [C_{center}, prev(r_i)])| == \theta$))
2. move to position returned by $\phi_{main}(r_i)$
3. **end while**

Description of Function $\phi_{main}(r_i)$: This function returns the next target position of robot r_i. First of all, the angle (α) subtended by the two direct neighbors on each other is calculated (Line-1).

Function $\phi_{main}(r_i)$:

1. $\alpha = |\angle([C_{center}, prev(r_i)], [C_{center}, next(r_i)])|$
2. **if**($\alpha == 2*\theta$) **then** return (mid(next(r_i), prev(r_i)))
3. **else if** ($\alpha < \theta$) **then** return r_i
4. **else**
5. $\lambda1 = |\angle([C_{center}, prev(r_i)],[C_{center}, prev(prev(r_i))])|$
6. $\lambda2 = |\angle([C_{center}, next(r_i)],[C_{center}, next(next(r_i))])|$
7. **if** ($\lambda1 == \lambda2 == \theta$ || ($\lambda1$!= θ && $\lambda2$!= θ)) **then**
8. $p_1 = CW_{position}(prev(r_i))$
9. $p_2 = CCW_{position}(next(r_i))$
10. **if** ($|\angle([C_{center}, r_i], [C_{center}, p_1])| < |\angle([C_{center}, r_i], [C_{center}, p_2])|$) **then**
11. return p_1
12. **else** return p_2
13. **end if**
14. **end if**
15. **end if**
16. **end if.**

If the angle α is equal to twice the characteristic angle, then a 2-stable configuration is possible and the robot simply moves to the angular mid-point of the direct neighbors (Line-2). If the angle (α) is less than the characteristic angle, then the robot r_i knows that neither 2 nor 1 stable configuration is possible and it stays silent (Line-3). If the angle (α) is greater than the characteristic angle, then the robot r_i evaluates, if making 1-stable configuration with one or both the neighbors can make them 2-stable or none of them can become 2-stable (Line-4 to 7). It selects the one which is closer and makes a 1-stable configuration with that robot (Line-10 to 12).

3.6 EXPERIMENTAL SETUP

In this section we present the high level design of the software system developed for localization, and communication in the multi robot system. Separate treatment has been given to the image processing and communication modules in Sections 3.6.1 and 3.6.2 respectively.

3.6.1 Image Processing based Multi-robot localization

Localization is the first step in multi-robot systems before any coordination algorithm can be implemented on them. It is necessary for the robots to know their absolute and relative positions with respect to each other and in the global reference frame. In this section, we have developed a software system for fast and robust localization of multi-robot system. Basically, we have emulated a Global Positioning System by means of simple image processing that is carried out by an external entity, i.e., an overhead video camera attached to a laptop or PC. It provides each robot the following information:

– Access to its own position and

– Access to the position of other robots in its own ego-centric coordinate system.

A special marker is designed and is attached on the top of the robots for unique identification and accurate localization. This marker uses four different colors, as shown in Figure 3.10. The position and orientation of up to five robots can be determined using an overhead camera with this marker. Use of additional colors can extend the limit on the number of robots, i.e., 16 robots with 5 colors, 25 robots with 6, and $(n\text{-}1)^2$ robots with n colors can be handled.

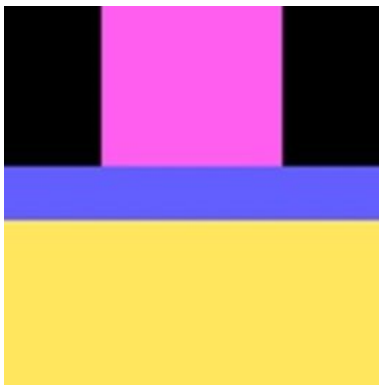


Figure 3.10 Rectangular marker with four different colors (the black color is subtracted)

The marker consists of a cyan strip in the center for recognition of a robot and its position estimation, along with two colored strips, one large and one small, for its identification as well as orientation. These two colored strips can have any three colors apart from cyan, hence

$3^2 = 9$ combinations are possible. The original image captured by the camera (the top view) after deploying multiple robots with fixed markers is shown in Figure 3.11(A).

Localizing multiple mobile robots is a two-step process. In the first step we need to recognize that a robot is present at a certain position in the shared environment and in the second step multiple robots are required to be identified uniquely using the attached markers. The steps in this process are described below:

- **Color segmentation**: Each pixel of the marker is classified into a color class based on RGB lookup tables for each of the four colors. The lookup tables are produced utilizing the algorithm identified in [Bruce 2000]. Figure 3.11(B) shows a 5-ary image, which contains the color label for each of the four colors on the marker as well as background for each pixel.
- **Blob identification (or connected component analysis)**: This requires labeling of each connected component. A connected component of each color of the marker is labeled using the algorithm mentioned in [Chang 2004]. Figure 3.11(C) shows the labeled image. Post-processing a filter is applied to remove too large or too small blobs, i.e., min and max thresholds for the area. Figure 3.11(D), shows the image after elimination of blobs. Once we have a list of all the blobs, i.e., connected components in the image, we can recognize, uniquely identify, and determine the position and orientation of all the robots. The algorithm for the same is given below:

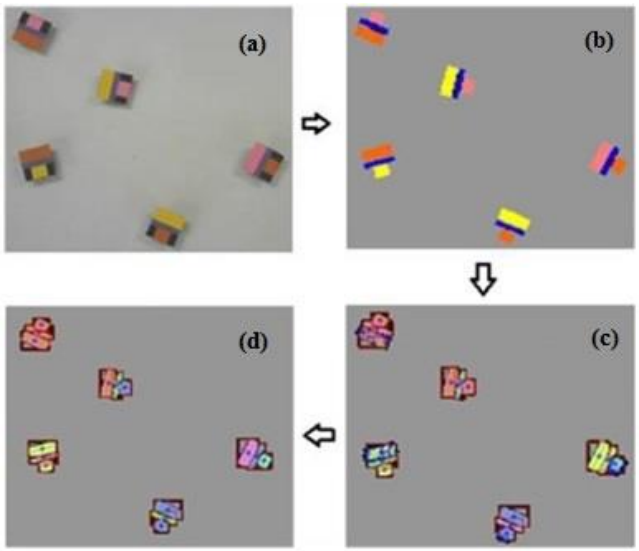


Figure 3.11 (a) Original image (top view of the camera), (b) 5-ary segmented image created using lookup tables for each of the four colors and black background (shown with grey color), (b) Markers labelled with blobs, (d) Small and large blobs are removed

Algorithm 3.2: Multi-Robot Localization

Each blob has three properties: (a) coordinates of the centroid (b) color and (c) area. The global list of blobs is split into three sub lists: (a) the list of cyan blobs (b) the list of large blobs and (c) the list of small blobs. For each cyan blob in the image, the following steps are carried out:

1. Determine the closest large blob and its color (henceforth referred to as LB).
2. Determine the closest small blob and its color (henceforth referred to as SB).
3. Using the two colors LB and SB, the index number of the robot is identified.
4. The robot's position is set as the coordinates (x, y) of the cyan blob.
5. The robot's heading is set as the direction vector from the small blob to the large blob. The calculations required to determine the robot's heading are shown below in eqs. 3.8 - 3.10. Figure 3.12 represents the same:

$$\vec{V}_x = \frac{X_{top\,\mathrm{Im}} - X_{CLB}}{\sqrt{\left(X_{top\,\mathrm{Im}} - X_{CLB}\right)^2 + \left(Y_{top\,\mathrm{Im}} - Y_{CLB}\right)^2}} \tag{3.8}$$

$$\vec{V}_y = \frac{Y_{top\,\mathrm{Im}} - Y_{CLB}}{\sqrt{\left(X_{top\,\mathrm{Im}} - X_{CLB}\right)^2 + \left(Y_{top\,\mathrm{Im}} - Y_{CLB}\right)^2}} \tag{3.9}$$

$$\theta = \begin{cases} a\cos(V_x), & V_y \le 0 \\ a\sin(V_y), & V_y \ge 0, \quad V_x \le 0 \\ a\sin(V_x) + \dfrac{3\Pi}{2}, & V_y \ge 0, \quad V_x \ge 0 \end{cases} \tag{3.10}$$

Where,

$0 \le \theta \le 2\Pi$,

X_{CLB} = coordinate of the center of large blob

Y_{CLB} = *y* coordinate of the center of large blob

$X_{top\,\mathrm{Im}}$ = topmost *x* coordinate of the image

$Y_{top\,\mathrm{Im}}$ = topmost *y* coordinate of the image

$\vec{V_x}$ and $\vec{V_y}$ are unit vectors representing the robot's orientation w.r.t image coordinates

6. All three blobs, cyan, large, and small are removed from their respective lists.

In steps (1) and (2), if no blob is found, or the closest blob is farther than a certain threshold, then that cyan blob is deleted from the list and the process continues. This happens rarely, when the robot is at the edge of the image and is only partially visible, or lighting errors have caused non-detection of some colors. In our experiment we have imposed movement restrictions on the robots, such that, they are not allowed to travel closer than 5 cm (empirically determined through camera calibrations) to the boundaries of the arena. Figure 3.13 shows the final image where multiple robots and their orientation has been determined. Since the camera is fixed at a pre-determined height converting pixels into meters, i.e., the coordinates of the pixel (X_{pixel}, Y_{pixel}) to the real world coordinates (X_{meters}, Y_{meters}) is straight forward. The only variable we have to consider is the resolution of the image frame referred to as $Width_{frame}$ and $Height_{frame}$. The dimensions of the 2D plane are previously known and are referred to as $Width_{arena}$ and $Height_{arena}$. Once we receive the position of the robots in pixel they are converted into real world coordinates using the formula given in eq. (3.11) and (3.12) below:

$$X_{meters} = \frac{X_{pixel}}{Width_{arena}} * Width_{frame} \tag{3.11}$$

$$Y_{meters} = \left(Height_{frame} - \frac{Y_{pixel}}{Height_{arena}} \right) * Height_{frame} \tag{3.12}$$

Usually the origin that is used for the image data type is the top left corner. Therefore, in eq. (3.12) the ratio of the pixel's Y coordinates and the arena's height is subtracted from the frame's height in order to change the origin to the bottom of the image.

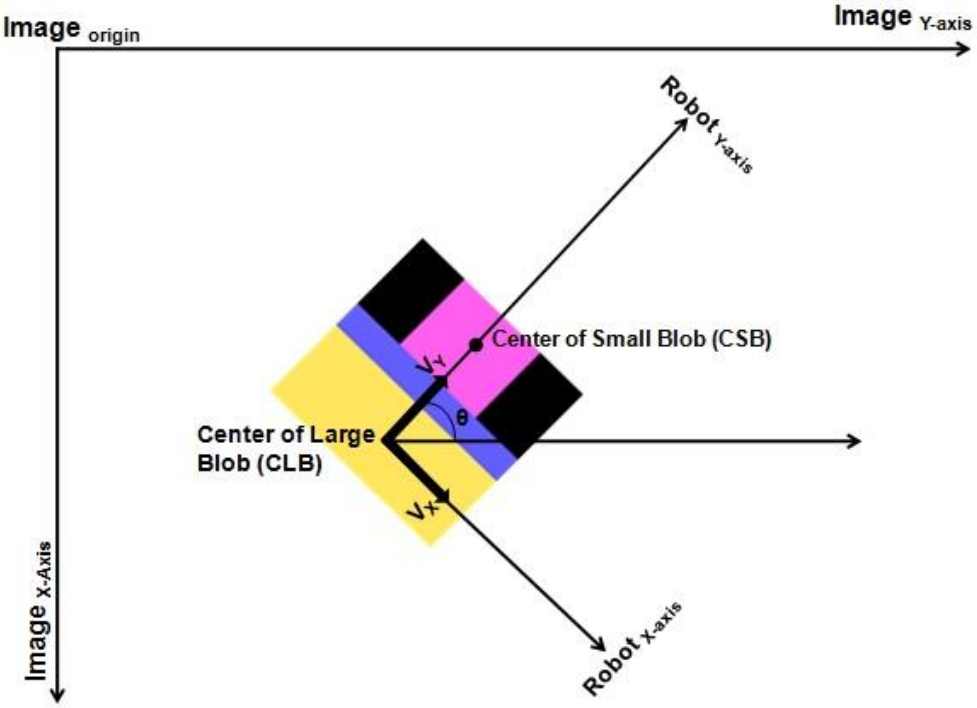


Figure 3.12 Global and local reference frame of robots

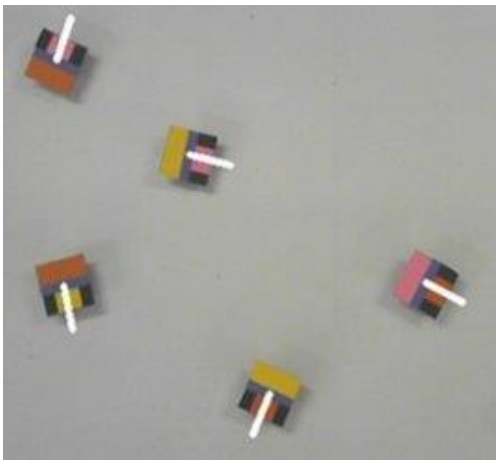


Figure 3.13 Final position and orientation of multiple robots

3.6.1.1 Computational Complexity of the Image Processing Employed for Multi-Robot Localization –

The suggested solution for multi-robot localization is a very inexpensive solution. The two main image processing steps of the algorithm, i.e., the color segmentation and the blob detection are both single pass algorithms, meaning, they access each image pixel only once. Identification of the robots requires finding the closest large and small blobs which is a nearest neighbor search problem having a complexity of *O(n)*. Scalability of the overall algorithm depends mainly on the size of the image. In an indoor environment where the area

is small and the robots used are less (typically not more than 25), it performs pretty well. But we can overcome this limitation by parallelizing some of the image processing steps on a rather high performance machine. This algorithm can be scaled for up to *n* robots by using the square root of *n+1* colors. Finally, we have successfully implemented a self-contained image processing module, which provides to each robot, absolute position and orientation of the robot itself and its peers using pull mechanism. Much effort has been spent in designing and implementation of robust software for reliable localization since the controller is going to generate the desired results only if it can trust the localization sub-system.

3.6.2 Multi-Robot Communication

Experiments are performed using e-puck robots [Mondada 2009]. The e-puck has several rich features but not much attention was paid to the wireless communication. The attached Bluetooth interface of e-puck only provides communication between a computer and the robot itself. In the past many extensions have been proposed to the basic robot's configuration to overcome this limitation. The range and bearing turret (e-RandB) proposed by [Gutiérrez 2009] allows short range communication (limited to 80 cms) with infrared. Apart from its limited communication range and low data rate the infrared signals are greatly influenced by the presence of obstacles and lighting conditions in the environment. To endorse cooperative robotics applications, in [Cianci 2006] Zigbee based communication is used. Network of robots can be formed with a flexibility of changing the communication range between 15 cm to 5 meters. This allows one to study the impact of communication range constraints on multi-robot applications in a laboratory setting. Besides several advantages of this extension, one primary limitation is that it is not commercial and has to be built from scratch. In this thesis, we have developed a self-contained software system which supports both asynchronous robot-to-computer and robot-to-robot communication. The following section describes how asynchronous communication is achieved.

3.6.2.1 Robot to Computer Communication

For a robot to computer communication star topology is suggested as shown in Figure 3.14. In client server architecture, the communication between the robots with the personal computer happens on e-puck's serial port over Bluetooth protocol. The software running on the computer sends appropriate commands to the robots and receives event acknowledgements from the robots. This is a traditional (synchronous) method which is based on blocking I/O, such that, the computer waits for a response from the robots before

issuing the command and vice-versa. Consequently, this mechanism is not really effective. The computer should be able to send the commands asynchronously to the robots without blocking. Similarly, the robots should be able to send acknowledgments without blocking. The Boost.Asio library [Radchuk 2013] which is a cross platform C++ library for low level I/O programming has been used for data communication between the e-puck robots and the computer. This library encapsulates essential building blocks for concurrency, networking, and other types of I/Os in C++. The default firmware (referred to as BTCom) of the e-puck robot is an open source software that provides simple text based interface to access the robot's hardware. We have modified BTCom and added support for asynchronous communication between robots and computer.

Figure 3.14 Multi-robot communication perceived as a star topology

3.6.2.2 Inter-Robot Communication

Each physical robot is associated with a virtual robot which is nothing but a network process with a unique identity (ID), as shown in Figure 3.15, henceforth referred to as Vrobot. The Vrobot with ID equal to N has a clockwise neighbor with ID equal to N+1 and an anticlockwise neighbor with ID equal to N-1. The role of the Vrobot is crucial, such that, they execute the STATE algorithm. All the active Vrobots pull the current system configuration after receiving a notification from the image processing based localization sub-system. They then compute the next target position of the physical robot associated with them and asynchronously send appropriate motion command to the physical robot. The Vrobots are supposed to synchronize with each other using the STATE algorithm and therefore they need to communicate with each other in an asynchronous manner. This is a challenging implementation because sockets in C do not provide intrinsic support for multithreading and there is no socket API in C++, i.e., std::socket does not exist. Therefore,

we have used a socket interface provided by Boost.Asio that provides a consistent asynchronous model for inter Vrobot communication over TCP sockets, as shown in Figure 3.15.

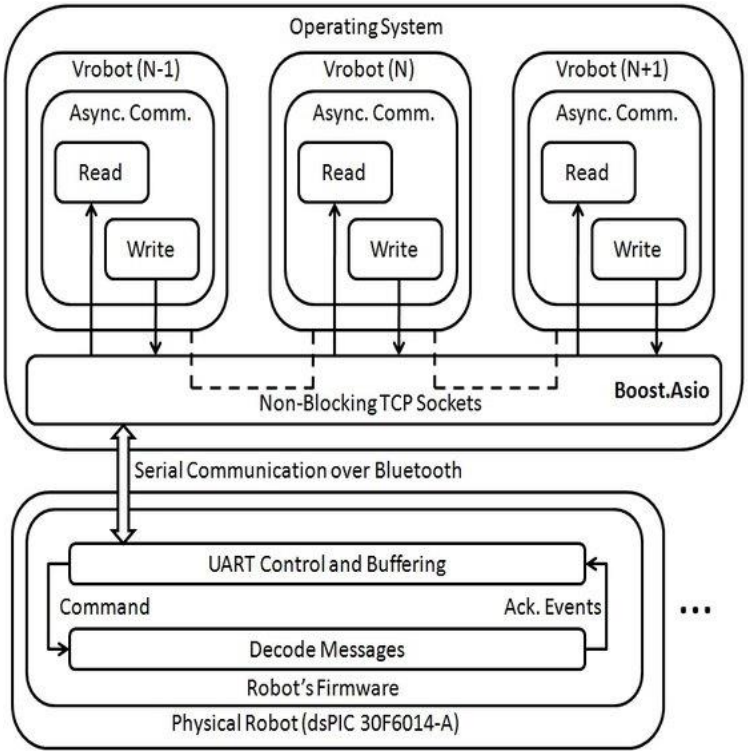


Figure 3.15 Proposed architecture for inter-robot communication using Boost.Asio library

3.7 RESULTS AND DISCUSSIONS

Russel and Norvig in [Russell 2014] have stated that "*an agent should subjectively be able tell how happy it is with its own performance but some agents would not be able to say anything on that front and some others will delude themselves*". Therefore, they insist on objective performance measurement wherein an external authority should establish a standard as to what it means to be successful in an environment and use it to measure the performance of an agent. In multi-robot systems this becomes challenging, because the performance of one robot has a direct impact on the performance of the team. In order to quantitatively compare the quality of the two algorithms for uniform circle formation by a set of mobile robots which act as a team, and follow a common protocol, two metrics have been chosen. They are defined below:

– Let A_α^i be the total number of activation steps performed by robot r_i when it meets the condition of convergence while executing algorithm α. Let $Total_{activations} = \sum_{i=1}^{N} A_\alpha^i$ (where, N = number of robots) be the total number of activation steps required by all the robots in the multi-robot system to form a uniform circular formation.

– Let T_α^i be the total time taken by robot r_i when it meets the condition of convergence while executing algorithm α. Let *MAX_TIME* denote the maximum time taken by the last robot that meets the condition of convergence, such that, $MAX\_TIME = \arg\max_{r_i \in R} T_\alpha^{r_i}$.

The activation cycle of the robot described previously in Section 3.5.4 is atomic and therefore we consider one activation cycle as one activation step. Execution of each activation step would consume the resources of robots, i.e., the robot carries out sense, compute and move operations which require some energy. Therefore, total number of activation steps for achieving convergence is a natural choice of comparison between the two algorithms. We cannot say that the algorithm that requires less number of activation steps is better, as the robots may not spend equal time in the execution of subsequent activation steps. Therefore, it becomes apparent to also measure the total time it takes for the last robot in the multi-robot system for achieving convergence. All other robots would have already achieved convergence by that time. Nevertheless, it has been observed by conducting several rounds of experiments that, if the number of activation steps is high, then the maximum time required for achieving convergence is also high.

Three different sets of results are obtained for circles with *small* (1 meter), *medium* (2 meters), and *large* (3 meters) diameters. In each set *five e-puck robots* are placed in three different initial organizations, i.e., *best placement*, *random placement*, and *worst placement*. The three different placement scenarios are shown in Figures 3.16 - 3.18 respectively. We have considered ten point robots sitting on the circle circumference for the sake of clarity and representation. In the *best placement* scenario, as shown in Figure 3.16, the initial placement of the robots itself brings the robot team into a circular formation which is nearly balanced, i.e., leaving a few robots all other robots are already subtending an angle of $2\pi/N$ on their clockwise and anti-clock wise neighbors. It can be seen in Figure 3.16 that robots (denoted by R) numbered 1, 2, and 6 should have been positioned at the midpoint, in terms of angular distance, of R=7 and R=10, R=9 and R=7, and R=5 and R=10 respectively. All other robots are correctly positioned and are in a balanced configuration. On the other hand, in case of

random placement scenario, as shown in Figure 3.17, the initial placement of the robots has many robots in an unbalanced configuration. In the worst placement scenario, the initial placement creates multiple clusters of the robots on a small arc of the circle. This is shown in Figure 3.18. Extensive experiments have shown that the size of the diameter, initial placement, and the chosen activation policy have significant impact on the total number of activation steps required and maximum time required by the robot team to achieve convergence.

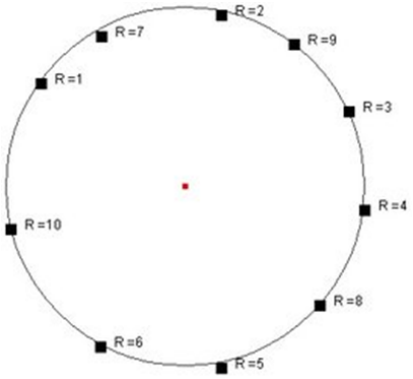


Figure 3.16 Best Placement

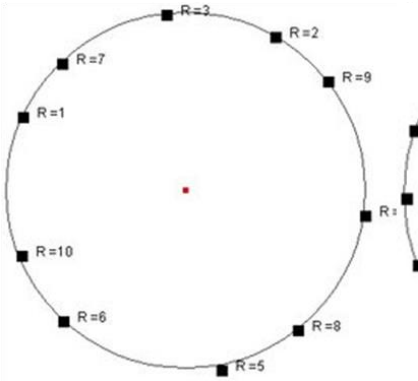


Figure 3.17 Random Placement

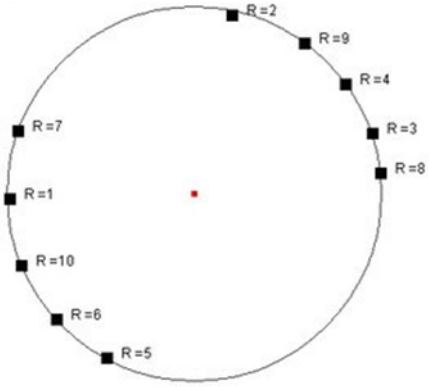


Figure 3.18 Worst Placement

In each placement scenario and for three different sizes of the circle we have conducted 25 runs of the two algorithms. Before we discuss specific results with respect to different sizes of the circle, certain common observations are made and are as follows:

(a) Referring to Table 3.1 to 3.3, it can be inferred that, irrespective of the size of the circle and the initial placement of the robots, if we increase the activation probability of the DK algorithm from $P_r = 0$ to $P_r = 1$, the percentage improvement (in terms of completion time) of the proposed (STATE) algorithm over the DK algorithm is reduced. This is because the activation policy of the DK algorithm is probabilistic. It has been discussed in section 3.5.4 that, in case of the DK algorithm, lower the probability of activation of robots, lesser is the number of robots active during each time instance, and hence the algorithm takes longer to converge. Nevertheless, when compared with DK algorithm the proposed (STATE) algorithm always performs better in all situations.

(b) Referring to Table 3.1 to 3.3, it can be observed that, fixing the activation policy of the DK algorithm, and increasing the size of the circle results in higher percentage improvement of the proposed (STATE) algorithm over DK algorithm.

(c) Referring to Table 3.1 to 3.3, it can also be observed that, irrespective of the activation policy of the DK algorithm, the percentage improvement of the proposed (STATE)

algorithm is highest when the initial placement of the robots is made as in the worst case (Figure 3.18).

We now discuss some specific cases with respect to different sizes of the circle:

(a) The average number of activation steps and the average maximum time to convergence for a circle with small diameter for DK algorithm (with different activation policies) and the STATE algorithm are shown in Figure 3.19 and Figure 3.20 respectively. From Table 3.1, it is evident that, for the three different placement scenarios i.e. the best, random and the worst, the percentage improvement achieved by STATE algorithm (in terms of number of activation steps and convergence time) when compared with different activation policies of the DK algorithm is significant for P_r = {0, 0.25} and not so significant for P_r = {0.5, 1}.

(b) Figure 3.21 and Figure 3.22 show the average number of activation steps and average maximum time to convergence required respectively for a medium size circle. It can be observed from Table 3.2 that, for the three different placement scenarios i.e. the best, random and the worst, the percentage improvement achieved by the STATE algorithm (in terms of number of activation steps and convergence time) is significant when compared with all the activation policies of the DK algorithm.

(c) Figure 3.23 and Figure 3.24 show the average number of activation steps and average maximum time to convergence required for the circle with large diameter respectively. Again in all the three different placement scenarios i.e. the best, random and the worst, the STATE algorithm performs better (in terms of number of activation steps and time) than all the activation policies of the DK algorithm. This observation follows from Table 3.3. It can easily be observed that the percentage improvement of the STATE algorithm over DK algorithm is significant in all the situations.

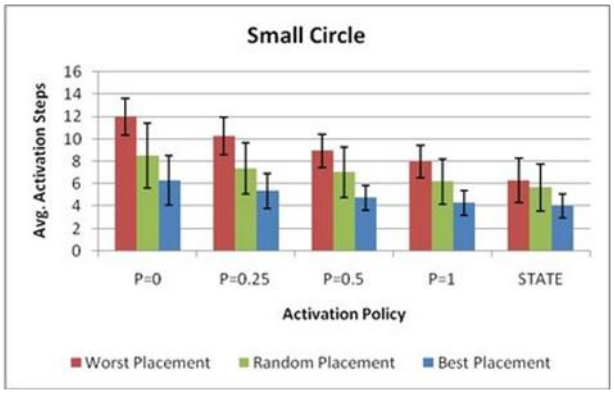


Figure 3.19 Average of *Total*$_{activations}$ for Small Sized Circle

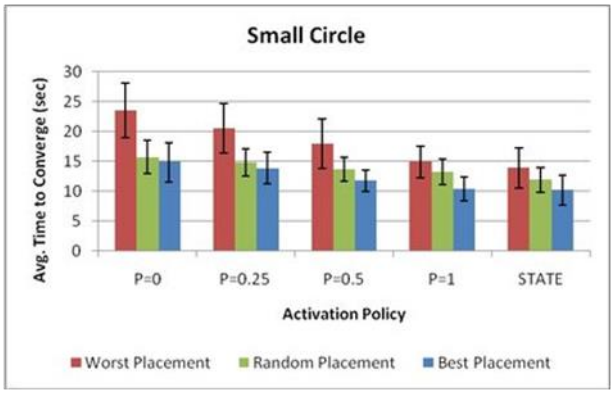


Figure 3.20 Average of *MAX_TIME* for Small Sized Circle

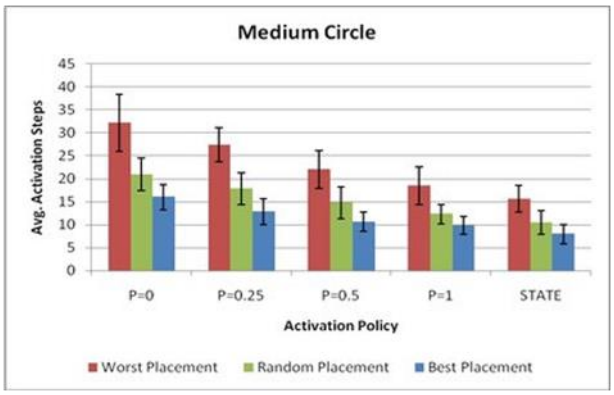


Figure 3.21 Average of *Total*$_{activations}$ for Medium Sized Circle

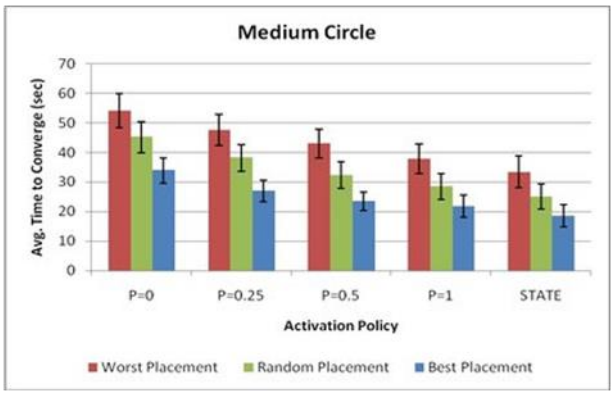


Figure 3.22 Average of *MAX_TIME* for Medium Sized Circle

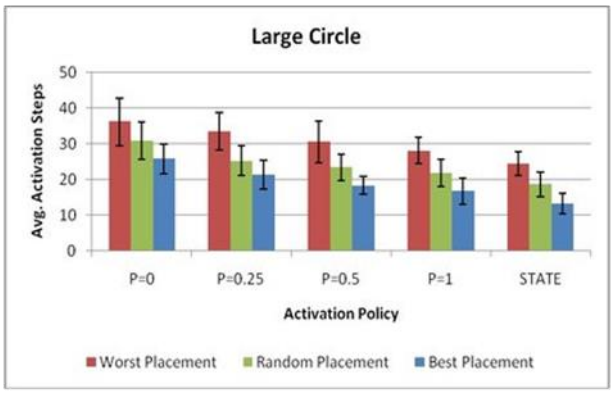


Figure 3.23 Average of *Total*$_{activations}$ for Large Sized Circle

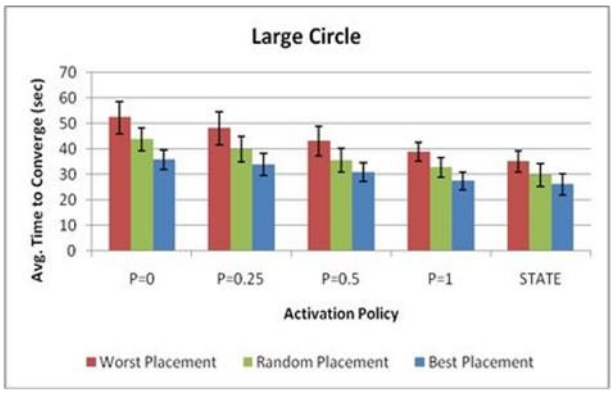


Figure 3.24 Average of *MAX_TIME* for Large Sized Circle

Table 3.1 Percentage Improvement of STATE Algorithm over DK Algorithm in case of Small Sized Circle (Diameter = 1 meter)

PLACEMENT	STATE vs.	PERCENTAGE IMPROVEMENT IN AVERAGE ACTIVATION STEPS	PERCENTAGE IMPROVEMENT IN AVERAGE COMPLETION TIME
BEST	DK (P=0)	32.37	24.29
	DK (P=0.25)	22.82	19.81
	DK (P=0.5)	8.12	9.38
	DK (P=1)	6.48	4.24
RANDOM	DK (P=0)	36.07	31.38
	DK (P=0.25)	24.62	26.3
	DK (P=0.5)	10.31	12.57
	DK (P=1)	8.38	6.68
WORST	DK (P=0)	47.15	32.78
	DK (P=0.25)	37.71	26.83
	DK (P=0.5)	27.66	18.32
	DK (P=1)	9.19	9.53

Table 3.2 Percentage Improvement of STATE Algorithm over DK Algorithm in case of Medium Sized Circle (Diameter = 2 meters)

PLACEMENT	STATE vs.	PERCENTAGE IMPROVEMENT IN AVERAGE ACTIVATION STEPS	PERCENTAGE IMPROVEMENT IN AVERAGE COMPLETION TIME
BEST	DK (P=0)	33.33	26.68
	DK (P=0.25)	26.03	22.85
	DK (P=0.5)	19.89	15.46
	DK (P=1)	12.82	5.04
RANDOM	DK (P=0)	39.55	31.87
	DK (P=0.25)	26.88	25.35
	DK (P=0.5)	20.61	17.36
	DK (P=1)	14.65	7.23
WORST	DK (P=0)	48.52	40.91
	DK (P=0.25)	38.28	32.33
	DK (P=0.5)	29.14	22.75
	DK (P=1)	19.12	9.83

Table 3.3 Percentage Improvement of STATE Algorithm over DK Algorithm in case of Large Sized Circle (Diameter = 3 meters)

PLACEMENT	STATE vs.	PERCENTAGE IMPROVEMENT IN AVERAGE ACTIVATION STEPS	PERCENTAGE IMPROVEMENT IN AVERAGE COMPLETION TIME
BEST	DK (P=0)	49.9	38.29
	DK (P=0.25)	37.81	29.79
	DK (P=0.5)	25.18	20.66
	DK (P=1)	14.93	11.83
RANDOM	DK (P=0)	50.25	44.44
	DK (P=0.25)	41.25	31.01
	DK (P=0.5)	28.97	22.59
	DK (P=1)	15.18	11.85
WORST	DK (P=0)	51.33	45.13
	DK (P=0.25)	42.83	34.27
	DK (P=0.5)	31.18	27.42
	DK (P=1)	20.95	14.86

Although, the STATE algorithm is found to perform better than the DK algorithm in terms of a smaller number of activation steps and lesser total time required for achieving convergence towards a uniform circular formation, as shown in the graphs of Figure 3.19-3.24 and from Table 3.1 – 3.3, the benefit of the strategy is not evident from the average values alone in certain cases i.e. of the small circle. Therefore, Wilcoxon signed rank test have been performed to verify whether the difference of the results of a given metric is statistically significant. For three different sizes of the circle (*small, medium, and large*) and for three different initial placements of the robots (*best, random, and worst*), we have recorded the values of 25 runs for both DK and the STATE algorithm. Probabilistic activation policy with P_r = {0, 0.25, 0.5, 1} is adopted for the DK algorithm. For the STATE algorithm order preserving activation policy is used. Since the data obtained is not normally distributed it makes sense to conduct the Wilcoxon signed rank test. Specifically, we performed a one-tailed test with the null hypothesis that the results of the two algorithms are statistically identical. The significance level of α=0.05 is considered to be compared with the *P*-values obtained from the experimental data. If the *P*-values obtained are less than α, the approaches are considered statistically different. The results of the Wilcoxon tests are presented in Table 3.4 (for the Activation Steps) and Table 3.5 (for Time). We have used the Wilcoxon signed

rank test calculator available from [SciStatCalc 2016]. It is evident from the results that the STATE algorithm performs better than Défago and Konagaya (DK) algorithm in terms of number of activation steps and total time required for achieving convergence. However, there exists an exception with the circle of small diameter. Referring to Table 3.4 and Table 3.5, for the Best and Random placement of the robots and the activation policy of P_r = 0.5 the *P*-values (shaded with blue color) are greater than 0.05 and the results are statistically identical, the null hypothesis is accepted. Similarly, for P_r = 1, for three different placements the *P*-values (shaded with grey color) are greater than 0.05 and the results are statistically identical and the null hypothesis is accepted. Considering a circle with small diameter and greater synchronization between the robots, we can conclude that there is no noticeable difference between the Défago and Konagaya (DK) algorithm and the proposed STATE algorithm. However, for the circle with bigger diameter irrespective of the level of synchronization in Défago and Konagaya (DK) algorithm, the STATE algorithm always performs better.

Table 3.4 P-values of Wilcoxon test for Activation Steps (α = 0.05, N=25)

	STATE Vs P_r=0			**STATE Vs P_r=0.25**			**STATE Vs P_r=0.5**			**STATE Vs P_r=1**		
Circle Size	***Best***	***Random***	***Worst***	***Best***	***Random***	***Worst***	***Best***	***Random***	***Worst***	***Best***	***Random***	***Worst***
Small	0.000615	0.001920	0.000018	0.004370	0.017022	0.000068	**0.074170**	**0.065452**	0.000383	**0.386925**	**0.489783**	**0.077767**
Medium	0.000012	0.000012	0.000012	0.000022	0.000027	0.000012	0.002700	0.000469	0.000025	0.009851	0.038620	0.031354
Large	0.000018	0.000018	0.000012	0.000018	0.000044	0.000044	0.000182	0.000153	0.000153	0.001644	0.009730	0.002031

Table 3.5 P-values of Wilcoxon test for Time in Seconds (α = 0.05, N= 25)

	STATE Vs P_r=0			STATE Vs P_r=0.25			STATE Vs P_r=0.5			STATE Vs P_r=1		
Circle Size	***Best***	***Random***	***Worst***	***Best***	***Random***	***Worst***	***Best***	***Random***	***Worst***	***Best***	***Random***	***Worst***
Small	0.000112	0.001303	0.000014	0.000446	0.011876	0.000081	**0.051087**	**0.201222**	0.007130	**0.756995**	**0.544910**	**0.275832**
Medium	0.000012	0.000012	0.000012	0.000029	0.000029	0.000016	0.000733	0.000733	0.000126	0.022988	0.026431	0.012814
Large	0.000016	0.000012	0.000012	0.000041	0.000036	0.000018	0.000194	0.000157	0.000065	0.000808	0.000665	0.000240

3.8 CHAPTER SUMMARY

In this chapter, several implementation issues with respect to different abstract assumptions considered in theoretical research in the area of multi-robot pattern formation are discussed. Approximate solutions to some of the assumptions have been proposed. A modified algorithm, STATE, is proposed and is shown to perform better than DK algorithm for solving the uniform circle formation problem. Both the algorithms are implemented on a real multi-robot test bed. A new framework for inter-robot communication is developed. It supports seamless asynchronous and non-blocking robot-to-computer, and robot-to-robot communications. The activation policy for achieving multi-robot synchronization in DK algorithm is probabilistic (discussed in Section 3.5.4). In our practical implementation of the DK algorithm a global clock is implemented for realizing the probabilistic activation schedule. It has been ensured that all the active robots finish executing the DK algorithm before a next clock tick is issued, to avoid race-condition. This resulted in more number of activation steps and longer time of convergence for the DK algorithm, because some robots have to remain in their wait/idle state for longer time. On the other hand, in the proposed approach in this thesis we have allowed the robots to explicitly communicate with their peers and therefore the robots are able to synchronize with their direct neighbors. It implies that, if a robot is active and is executing the STATE algorithm its direct neighbors stay silent.

Furthermore, contrary to the DK algorithm, in the STATE algorithm the robots directly move to the mid-point of their neighbors and thus achieve convergence in lesser number of activation steps. This also results in a smaller time to convergence. Finally, it is argued that proper coordination among the robots improves the efficiency of the task at hand. In chapter 4, research has been conducted on multi-robot coordination in the context of unknown terrain coverage.

CHAPTER 4

AN EFFICIENT APPROACH FOR ONLINE MULTI-ROBOT TERRAIN COVERAGE

4.1 INTRODUCTION

Multi-robot terrain coverage has gained considerable attention during the last two decades. The activity requires the robots to traverse the entire terrain for complete coverage. Various tasks requiring terrain coverage are sometimes monotonous (floor cleaning), time consuming (harvesting) and/or life threatening (inspection of hazardous terrains). Hence, automation of these tasks is highly desirable. Online algorithms for terrain coverage [Yamauchi 1998], [Wurm 2008], [Sheng 2006] do not rely on the cognition of the terrain. Therefore, the robots incrementally construct their trajectories and maps in real-time. An efficient coordination strategy results in faster coverage [Burgard 2005]. However, it also creates several challenges, as the robots have to execute many tasks in parallel, such as obstacle avoidance, mapping, localization, path planning, communication with peers, etc. Moreover, the very nature of the deployment of mobile multi-robot systems requires the robots to be simple in terms of their computational, sensing, storage, and communication capabilities. Knowing that the plan of one robot influences the plan of other robots intrinsically it becomes important for the robot team to properly coordinate.

This work presents an efficient approach for designating to each robot a set of frontiers. A path planning method which allows the robots to cover the set of frontier cells allocated to them in a less redundant manner is proposed. Empirical results both in simulation and on a multi-robot test-bed demonstrate that the proposed approach achieves faster completion of coverage and higher utilization of robots when compared with other state of the art approaches. The remainder of this chapter is organized as follows. There is a very thin line of difference between coverage and exploration which is explained in Section 4.2. A brief discussion on a set of approaches which are representative of the state-of-the-art on terrain coverage is presented in Section 4.3. These approaches are experimentally compared with the proposed approach. The motivation of this research and our contributions are presented in Section 4.4. The potential targets for terrain coverage are described in Section 4.5. Preparation of task subsets using a well-known unsupervised data clustering algorithm i.e.

the *K*-means algorithm is discussed in Section 4.6. The *frontier clusters* thus obtained are allocated to individual robots using an optimal task assignment algorithm i.e., the Hungarian method, this process is described in Section 4.7. The proposed approach is detailed in Section 4.8 followed by experimental results and analysis which are discussed in Section 4.9. Section 4.10 presents the chapter summary.

4.2 COVERAGE VERSUS EXPLORATION

Before we start our discussion on some of the state-of-the-art approaches it is important to bring out the distinction between the task of terrain coverage and terrain exploration. Terrain coverage requires the robots to completely traverse the unknown terrain in lesser time while exploration requires the robots to build a quality global map of the unknown region. In exploration the robots after scanning a region simply head towards unexplored territories, they are not required to traverse the whole region. Although, they are two fundamentally different applications but there is a lot commonality between them in terms of operations performed by robots which are as follows:

- The map of the terrain is not known to the robots beforehand.
- Approximate cell decomposition is used to decompose the terrain in equal sized grid cells. This is not the only method of terrain decomposition but we have limited our scope to frontier exploration techniques alone.
- The robots should be able to localize themselves. In the absence of a global map simultaneous localization and mapping (*SLAM*) techniques [Dissanayake 2001], [Montemerlo 2003] are used for building the map.
- The robots plan their next move based on the map information generated locally using their sensors and/or received from their peers. In that sense the robots fuse the useful map information obtained from other robots for the purpose of navigation.
- The robots communicate with each other, to exchange information regarding the local instance of the global map known to the robot at that point of time and sometimes to synchronize their actions. They may also share other information like their pose, current task assigned to them and its completion status, and their health status.

One important difference between the two applications is that, in the process of exploration the robots have a sensing radius that is higher than one cell and for the terrain coverage task

the robots are able to sense only the eight surrounding cells from the robot's position. Before this argument is accepted as a fact it is essential to discuss some of the recent state of the art multi-robot exploration approaches. Brick&Mortar [Ferranti 2007], RAPID [Ferranti 2009], and BMI [Andries 2013] are three different ant based approaches in which the environment is divided into a grid of square cells. The size of the cell depends on two things (a) the distance measured by the range sensors (r_{sense}) and (b) the communication range (r_{comm}) of the agents. In [Ferranti 2009] it is assumed that the agent sitting at the center of some cell *c* will be able to cover *c* entirely and therefore the size of the cell *c* i.e., *x* should be less than or equal to $\frac{2*r_{sense}}{\sqrt{2}}$ and the agent is able to communicate within eight cells surrounding *c*, therefore, the communication range of the robot is less than or equal to $\frac{2*r_{comm}}{3\sqrt{2}}$, as shown in Figure 4.1.

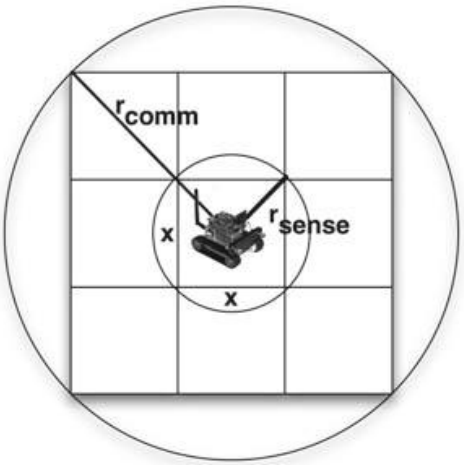


Figure 4.1 Sensing (r_{sense}) and Communication Range (r_{comm}) of Robots [Ferranti 2009]

RAPID is an extension and BMI is an improvement over Brick&Mortar. These three are suggested as exploration approaches by the authors. Yet the robot's coverage is limited to only one cell in which it is currently located and it is able to sense the status of eight surrounding cells by probing the smart tags lying in those cells. Therefore, if the sensing range of the robots is restricted to one cell all exploration algorithms will transform into coverage algorithms. We argue that terrain coverage is also a form of exploration with one additional requirement that the robots have to physically traverse whole of the free space and which is a single connected component. We have compared some of the state of the art approaches of exploration [Yamauchi 1998], [Burgard 2005], and [Puig 2011] (restricting the sensing range of robots to one cell) with the approach presented in this chapter.

4.3 DISCUSSION ON REPRESENTATIVE APPROACHES

In this section, a representative set of important multi-robot exploration approaches which are compared with the proposed approach are discussed. Three highly praised techniques have been considered:

(a) Pure frontier exploration [Yamauchi 1998] - It is the most basic form of frontier exploration wherein the robots do not coordinate with each other. They just exchange their map data with their peers and greedily select the target frontier cells.

(b) Coordinated exploration [Burgard 2005] - This is an extension of Pure Frontier Exploration. The robots coordinate with each other for the selection of the frontier cells. However, the coordination mechanism is not very sophisticated.

(c) Dispersion based exploration [Puig 2011] - This algorithm exhibits stronger form of coordination than the other two. The unknown region is partitioned and optimally assigned to individual robots for further exploration.

These techniques are different from each other in the level of multi-robot coordination. It is worth comparing these methods within a common framework and evaluates how fruitful they will be for the purpose of online terrain coverage task. This helps us in identifying the advantages and disadvantages of each method so that the most suitable method can be selected for the said task. The working and the chief characteristics of each method is addressed in the rest of this section.

4.3.1 Pure frontier exploration

A popular technique referred to as frontier-based exploration for a single mobile robot is proposed in [Yamauchi 1997]. The border between the known and the unknown region is referred to as a frontier. The map unfolds as the robot moves to explore the nearest frontier. Formally the process is described as follows: Suppose $F=\{f_1, f_2, f_3, ..., f_k\}$ is the collection of frontier cells currently being evaluated. The current position of a robot r_i is $pos(r_i)$. The nearest frontier, say *nf*, is the one that can be reached in least cost from $pos(r_i)$ and is obtained as follows:

$$nf = \underset{f_i \in F}{\arg min}\, path(f_i, pos(r_i)) \tag{4.1}$$

$path()$ is a function that returns the length of the shortest path from some cell c_i to c_j. In this case both the nearest frontier and the shortest path are determined using Dijkstra's

algorithm [Dijkstra 1959]. The robot then reaches the nearest frontier and continues exploration. The procedure is repeated until no more frontiers are visible. This technique has been extended to multiple robots in [Yamauchi 1998]. Each robot has its own local instance of the global grid map. Every robot after reaching its nearest frontier observes the environment using its sensors and generates a local evidence grid. The robot integrates its local impression with its own local instance of the global grid map. Also, the robot broadcasts this information to all the other robots. The other robots receiving these broadcasts stores them and integrate this information in their own global grid map after reaching their destinations i.e., the frontiers cells chosen by them. Based on the knowledge of the global grid map the robots avoid visiting already explored frontiers. From the experimental perspective this method sequences the frontier cells by clustering those frontier cells which are in the immediate neighborhood. Small frontier sequences are ignored. The mid-points of the frontier sequences are selected as potential target for exploration. Let us call them target frontier cells. Navigation cost starting from a robot's position to all the target frontier cells is propagated in the multi-robot system. Euclidean distance cost is assumed between frontier cells. Infinite cost is associated to those cells which correspond to obstacles and unknown cells and some penalty is associated to those cells which are too close to obstacles for the robot to derive and follow a safe path. Each robot selects a target frontier cell with the minimum cost path (*mcp*) found after backtracking. The low-level planner of the robot makes the robot follow the *mcp*. The planner executes again and reassesses the path in three situations (a) after a specified time (b) when the robot reaches its destination and (c) when the path is blocked by an obstacle. This revolutionary work has a limitation that it allocates the frontiers to the robots, which may lead to improper utilization of resources by assigning the same frontier to multiple robots. This is due to the fact that while selecting a particular target frontier cell *x* only the *mcp* to *x* is considered and not the information gain that can be obtained by visiting *x*. The robots stay close to each other confined within the communication range and disperse locally. Despite the aforementioned limitation, the frontier-based exploration technique has become a de-facto standard for multi-robot exploration.

4.3.2 Coordinated exploration

It is a decision theoretic approach which extends [Yamauchi 1998] so that the robots coordinate with each other for faster completion of the exploration task. A function is defined that trades of the utility with the cost of reaching a particular target frontier cell. The

cost is nothing but the length of the shortest optimal path from the robot's position to the target frontier cell. The utility of a target frontier is the area that can be discovered when the robot arrives at it. The utility of target frontier cell *x* is lesser when the target frontier cells in the neighborhood of *x* are assigned to other robots. The target frontiers cells with the maximum utility are chosen by the robots. This ensures that the robots select the target frontier cells that are far from each other. Following is the formal summarization of the whole idea:

Let $F=\{f_1, f_2, f_3, ..., f_n\}$ be a set of target frontier cells. Let f_r be a target frontier cell for some robot *r* with a position `pos(r)`. Let `cost(`f_r`,pos(r))` be a function that returns the cost of robot *r* reaching the target frontier cell f_r. The cost is calculated as a ratio of the shortest optimal path from the position of the robot `r` to the target frontier f_r and the maximum of the shortest optimal path length of robot *r* reaching all the other target frontiers $F'=\{F - f_r\}$. The cost function is defined below:

$$cost(f_r, pos(r)) = \frac{path(f_r,\ pos(r))}{\max\limits_{f_j \in \{F - f_r\}} path(f_j, pos(r))} \tag{4.2}$$

The `path` function is defined above in the Yamauchi's algorithm. The utility of a robot *r* reaching the target frontier f_r is calculated as follows:

$$util(f_r, r) = \sum_{Z \in \{R - r\}} p(f_r, Z) \tag{4.3}$$

Here *R* is the set of all robots including robot *r* and *Z* is the set of all robot excluding robot *r*. Function *p* returns the distance between the target frontier cells chosen by other robots and the target frontier cell f_r selected by robot *r*. The value of *p* is evaluated as follows:

$$p(f_r, Z) = \begin{cases} dist(f_r, f_Z^i)/\lambda, & if \quad dist(f_r, f_Z^i) < \lambda \\ 0, & otherwise \end{cases} \tag{4.4}$$

where f_Z^i is the target frontier cell assigned to robot $i \in Z$, λ is a parameter that models the robots sphere of influence. Now the benefit *b* obtained by robot *r* when it selects the target frontier cell f_r is given below:

$$b(f_r, r) = util(f_r, r) - cost(f_r, pos(r)) \tag{4.5}$$

Each robot sends a bid (the benefit) for each target frontier cell to all the other robots. Each robot selects a target frontier cell which has a maximum bid for itself. Finally, the robot plans a path to the chosen target frontier cell. Although the overlap between the robots reduces and the approach tends to minimize the completion time but it confines the robots

locally within the sensing range i.e., the robots disperse locally until there is no overlap in the newly discovered region and therefore is similar to Yamauchi's algorithm. Moreover, the algorithm does not scale well even in a graph of moderate size. Another shortcoming of this work is that it does not try to optimize the allocation of the target frontier cells to the robots.

4.3.3 Dispersion based exploration

For the purpose of exploration dispersion of robots is particularly important. In fact, dispersion remains at the heart of most of the exploration algorithms. The more the robots are dispersed better will they be able to discover new areas and thus complete the exploration faster. This is the main theme of the above algorithm. Instead of allocating the frontiers cells (which are many in large maps) using auctions the cells in the unknown region of the occupancy grid map are clustered into disjoint regions using the *K*-means algorithm. The number of clusters is equal to the number of robots. The initial seeds for the *K*-means algorithm are chosen randomly. Euclidean distance to the centroid of a region from the current position of the robot is considered for computing an optimal assignment of robots to distinct regions using linear programming. The next task is that of leading the robot to their assigned regions. For each robot-region pair if there is a direct path from the robot's position to the region's centroid via the free space, the region is accessible to the robot it is assigned to. Otherwise the region is inaccessible. In that case the robot selects a frontier cell in the accessible region which is nearest to the region assigned to it. Separate distances are defined for reaching to both accessible and inaccessible regions as follows:

- Let us denote the accessible region by A and the robot by r. The distance from r to A is the distance from r to the closest cell $c \in A$ and is a shortest real path distance (minimum cost path in the previously discussed approaches) from r to all frontier cells in the neighborhood of c. If F_c is the neighborhood of c then the distance is calculated as follows:

$$d(r, c) = \min \{ p(r, f) \mid f \in F_c \}, c \in A \tag{4.6}$$

  where function p returns the shortest path.

- Let us denote the inaccessible region with I. For region I geometric distance is defined from r to c. If there is an obstacle on a direct line connecting r and c a penalty ρ is added to the distance calculation as follows.

$$d(r, c) = g(r, c) + \rho, \ c \in I \tag{4.7}$$

- The final objective is to find a robot-region pair which is the minimum of the distances to all regions (say *AR*) as shown below:

$$d(r, AR) = \min\ d(r, c),\ c \in A\ or\ c \in I \tag{4.8}$$

After the region assignment goals are assigned to the robots. The robots which are assigned to inaccessible regions have higher precedence. The distance between a target frontier and a robot is calculated as follows:

$$d(r_i, f_i) = \begin{cases} \delta(r_i, f_i) + \rho(f_i, nc_r^i), & f_j \in A \\ 0, \quad otherwise \end{cases} \tag{4.9}$$

Where nc_r^i is the nearest cell of the region assigned to robot r_i. The quantity $\rho(f_i, nc_r^i)$ is equal to the sum of geometric distance between f_i and nc_r^i i.e., $g(f_i, nc_r^i)$, penalty ρ_1 if there is an obstacle between f_i and nc_r^i i.e., $\rho_1(f_i, nc_r^i)$ and if the frontier f_i is already chosen by some other robot another penalty ρ_2 i.e., $\rho_2(f_i)$ is added. When the robots arrive at their destinations Burgard's algorithm [Burgard 2005] is used for the purpose of exploration within the region. The main difference with the [Yamauchi 1998] and the [Burgard 2005] is that this approach targets global dispersion. Global dispersion is successfully achieved by penalizing the robots when they choose frontiers that do not belong to their assigned regions or are chosen by other robots. Although, this approach is successful in achieving global dispersion and claims to complete the exploration faster by reducing the regional waiting time, it has few significant limitations. The limitations are:

- The *K*-means algorithm uses Euclidian distance for clustering. The quality of the clusters obtained in a free space (the region without any obstacles and walls) and the region with obstacles of arbitrary size, shapes and walls will be completely different. Suppose the map is a known map, such that, the planner has complete knowledge of the geometry of the obstacles and the walls in the environment. In this situation instead of using traditional *K*-means algorithm, Geodesic *K*-means clustering [Asgharbeygi 2008] can be used. This only allows the grid cells in neighborhood i.e., those that are not separated by obstacles to be clustered together as shown in Figure 4.4. Since the map is unknown and the size, shape and location of the obstacles and the walls is also not known a prior, geodesic distance cannot be used. This surely results in producing clusters of grid cells that are separated by obstacles as shown in Figure 4.3. As a result, the robots may have to travel much longer distances. This

issue is not addressed by the authors and therefore all the three claims of minimum completion time, minimum total path length travelled by the robots, and minimum variance of regional waiting time are questionable.

- The primary reason for the above situation is that the robots are directly sent to explore unknown territories therefore they were not able to exploit the global knowledge which is getting generated as they explore and discover walls and obstacles. Since no re-planning or re-clustering is done the robots have to travel longer distances.
- In order to guarantee the completion of exploration task either the individual robots will have to communicate their task completion status and the map information to all the other robots, like [Yamauchi 1998] and [Burgard 2005] or it should be a centralized algorithm. The algorithm sends the robots in far off regions thus it requires long range inter-robot communication. In its present state the algorithm is centralized as it states nothing about who is running the clustering algorithm.
- The algorithm is not robust to failure of robots. Let us assume that robots fail with some random probability and one robot survives till the end to finish the exploration task. This approach will have to repeatedly cluster the unknown region every time a robot fails (let us call it iteration). In large maps the volume of frontier cells to be clustered remains high and therefore the algorithm progresses slowly.

At this point it is very important to state that when these algorithms are employed for the purpose of online terrain coverage they are required to visit each and every frontier cell discovered in the process of execution until no more frontier cells are visible. Therefore, the algorithms are improvised in the manner that they do not ignore any frontier cell. Moreover, each frontier cell is a potential target for coverage and continues to remain a frontier until it is not traversed by a robot. Each frontier cell gets equal opportunity for it to be selected by some robot for coverage. The path planning algorithm executes only when the frontiers are more than two grid cells away from the robot's position.

4.4 RESEARCH MOTIVATION AND CONTRIBUTION

The main motivation behind this work comes from the fact that the potential of reuse of algorithms designed for terrain exploration has not been considered for the purpose of terrain coverage. But it is possible to improvise exploration algorithms and achieve online terrain coverage. The main limitations of previous approaches are:

(a) Many previous approaches which are based on frontier exploration [Burgard 2005, Sheng 2006, Zlot 2002] allow the robots to disperse locally within the sensing range of the range sensors. This increases the overlapping sensing impression of the robots which in turn increases the chance of robots selecting nearby target frontiers for exploration. This produces difficulty in managing the exploration process due to the following reasons:

- Interference of sensors.
- The robots spend a lot of time in collision avoidance with other robots thus affecting the speed of the exploration algorithm.
- In a larger terrain some portion of the terrain will be explored much later than the portion from where the robots have actually started exploring.
- These approaches are *exploitatory* in nature. The robots tend to exploit their knowledge to its maximum and explore newer regions in a conservative manner. As a result, exploration becomes slower.

(b) Approaches which are more *exploratory* in nature target global dispersion. The environment is partitioned into bigger segments which are then assigned to the robots for mutually exclusive exploration or coverage. Various techniques are used for the purpose of partitioning including data clustering algorithms like *K*-means [Stachniss 2006], [Wu 2010], [Puig 2011], graph coloring [Carvalho 2013], graph partitioning [Fazli 2010], and Voronoi partitioning [Wu 2007], [Hungerford 2016]. These methods have obvious advantage over the pure frontier based approaches in terms of faster task completion time [Wurm 2008] but also have many challenges:

- *Fixed partitions*: the robots plan longer paths as they do not exploit the knowledge of the structure of the environment that gets generated when they explore or cover the environment. Moreover, the motion plans they use result in redundant coverage.
- *Fault tolerance*: handling robot(s) failure has not been addressed by the above approaches. To detect the same either long range communication is required or the robots should develop consensus and go to a rendezvous point periodically.
- *Re-planning*: is an essential component of a good algorithm. None of these approaches specify strict time criteria for re-planning.

An efficient algorithm should be both *exploratory* as well as *exploitatory* in nature. In case of robot failures or when substantial new information is available, the plan should be revised. To overcome some of the limitations of the above discussed algorithms the proposed approach finds an appropriate motion plan for the robots based on the context of already covered frontiers. Dispersion of robots is vital for efficient coverage and is an emergent behavior. In that sense the robots are not sent to cover the unknown region immediately. In fact, they exploit the knowledge of the known frontier cells first. As the map unfolds and more frontiers are discovered the robots start exploring aggressively. The efficacy of the proposed approach is tested in simulation and on a multi-robot test-bed. The suggested approach performs better than some state of the art approaches [Burgard 2005], [Puig 2011], [Yamauchi 1998] and is well suited for an indoor environment for the purpose like floor cleaning etc.

4.5 TARGETS FOR TERRAIN COVERAGE

For multi-robot terrain coverage, most of the approaches suggested in the literature advocate generating a set of coverage targets, intelligently club these targets together to form task subsets, and assign these task subsets to the robots [Viguria 2007, Viguria 2008]. A coverage target is nothing but some specific location(s) in the environment which is to be traversed by a robot. In order to cover a target, the robot has to physically traverse its location and perform some task, for example, cleaning the target.

The representation of the environment has a profound impact on the process of generation of coverage targets. An occupancy grid map discretizes the environment into grid cells [Elfes 1989] and is a popular representation. The state of each cell, of being occupied or free, is estimated by the application of binary Bayes filter. In the beginning each cell is initialized as unknown. It is only after the *POSTERIOR* measurements are integrated that the occupancy of each cell is determined and is characterized as either occupied (if the sensed area contains the obstacle) or free (if it is a free space). One of the earliest works, by Brian Yamauchi [Yamauchi 1998] makes use of the occupancy grid based maps for robot exploration and takes into account the sensor measurements defining the region of the map that is sensed and what is still unknown. The cells that sit on the boundary of the known and the unknown regions are referred to as *frontier cells* and are potentially good targets for the robot to proceed for exploration. Moreover, each cell in the exploration task can be in one of the four states i.e., *unknown, explored, obstacle*, and *frontier*. Figure 4.2 gives an illustration of frontier based exploration using multiple mobile robots. It is already known that while

exploring the robots scan their surroundings and all the unknown cells within the sensing range of the robots are declared to be explored and are eliminated from being an exploration target for the robots. On the other hand, terrain coverage requires the robots to physically traverse the entire free space, be it in the known or the unknown region and therefore an additional flag variable, henceforth referred to as *visited*, is attached with every cell in the free space. The *visited* flag of a particular cell is set to true when the robot physically traverses it. As the robots continue with the coverage task, they sense their surroundings discovering new frontiers. The state of all the frontier cells which have been discovered in the past and are yet to be covered by the robot changes to *explored* but their *visited* flag remains false. These cells will continue to exist as potential targets for coverage.

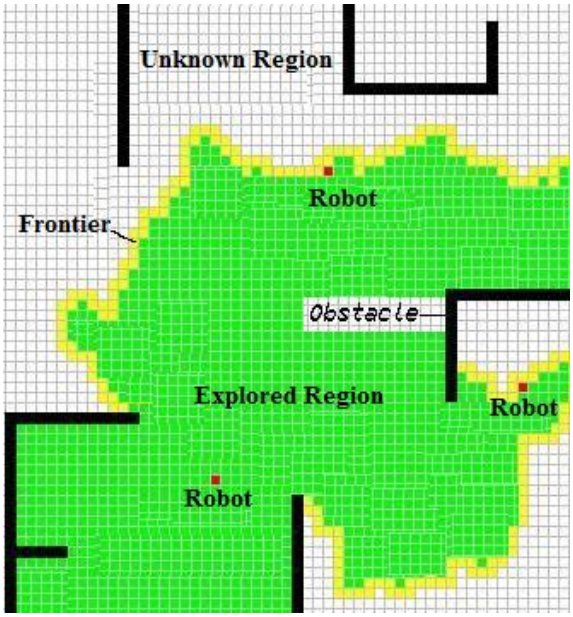


Figure 4.2 Frontiers based exploration

4.6 PREPARATION OF TASK-SUBSETS

In continuation to the discussion that was carried out in Section 4.3 it is evident that instead of allocating the frontier cells to the robots if the environment is decomposed into multiple segments/ regions and then assigned to the robots, the terrain coverage task can be achieved in more efficient manner i.e., with reduced time of task completion and reduced overlapping coverage. It is mainly because the robots would be doing coverage in mutually exclusive regions. Let us refer to these segments as task subsets. In this thesis, for the preparation of task subsets, the *K*-means algorithm is used for clustering the known frontier cells, hereafter referred to as *frontier clusters*. The *frontier clusters* thus obtained are allocated to the robots using an optimal task allocation algorithm (discussed in Section 4.7). We have also

discussed previously that conventional *K*-means algorithm uses *Euclidean* distances, that causes undesirable effects when frontier cells are separated by obstacles. It results in robots travelling longer distances to visit cells in their allotted *frontier cluster*s. This situation is described in Figure 4.3. It can be seen that three *frontier clusters* are formed wherein the frontiers cells are separated by obstacles and are far apart. The trajectories of two robots *R1* and *R3* that are allocated these *frontier clusters* are shown in green and dark blue colors. Another issue is that since only three *frontier clusters* are formed they are allocated to three robots *R1, R2,* and *R3*. The other two robots *R4* and *R5* are not allocated anything and remain idle, which is a waste of resources. In order to ensure that the frontier cells in a particular *frontier cluster* are not separated by obstacles and are close to the robot to which this cluster is allotted, geodesic distance [Asgharbeygi 2008] is used in the proposed approach. The effect is clearly evident in Figure 4.4. Five different *frontier clusters* are formed and are allocated to all the five robots. The trajectories of all the robots are shown in different colors. The task allocation mechanism is explained in the next section.

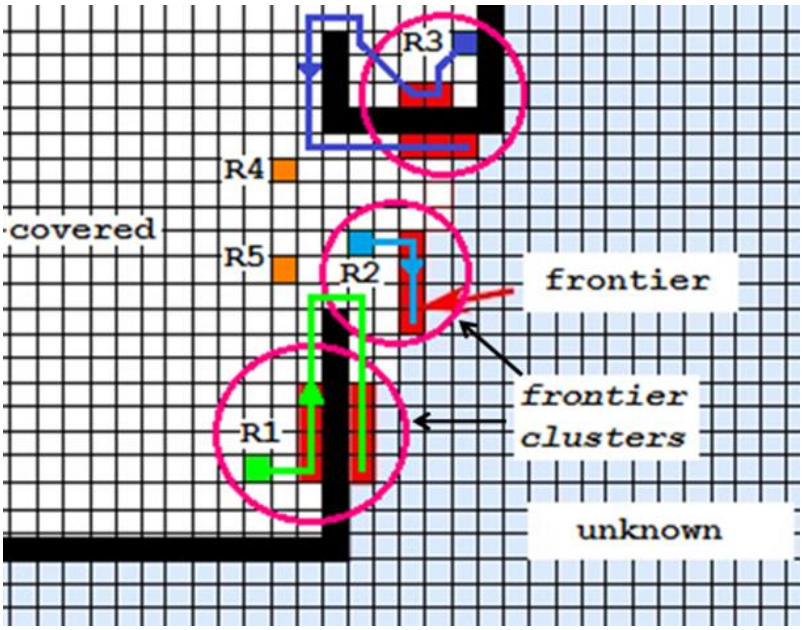


Figure 4.3 Frontier clusters formed with the application of conventional *K*-means algorithm

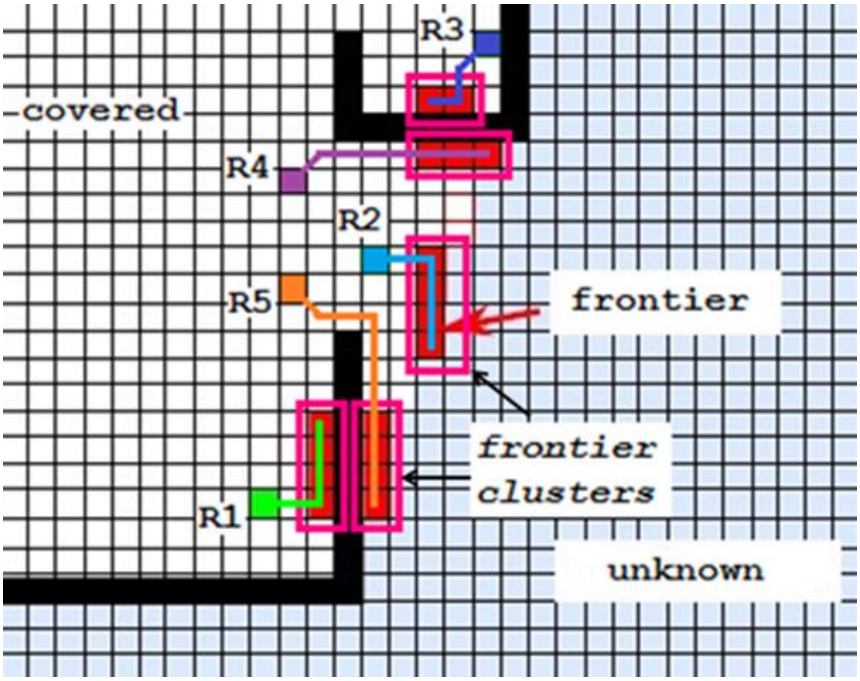


Figure 4.4 Frontier clusters formed with the application of Geodesic *K*-means algorithm

4.7 THE HUNGARIAN METHOD FOR TASK ASSIGNMENT

After the *frontier clusters* are obtained they are required to be allocated to the robots. Previously iterative and batch assignment approaches have been used for this purpose. In iterative method, the robots are assigned targets in a sequential manner one after the other [Burgard 2005]. On the other hand, in the batch assignment the robots are optimally assigned the targets at once [Ko 2003]. In our approach, one of the famous combinatorial optimization algorithm known as Hungarian method [Kuhn 1955] for task assignment is used. Given a fixed cost matrix, the Hungarian method optimally allocates a set of jobs to a set of machines. The final assignment always results in minimizing the overall cost. This algorithm takes as input a square matrix ($n \times n$) representing the cost of assigning n jobs to n machines.

The steps of the algorithm are summarized below:

1. First of all, a reduced cost matrix is computed by subtracting the minimal element i_{min} of the i^{th} row from all the elements of the i^{th} row. After that the same operation is carried out on all the columns.

2. Next, it is required to find out the minimal number of lines, both horizontal and vertical (referred to as *cut-lines*), that are sufficient to strike of all the zeros in the cost matrix. If exactly n lines are required, then an optimal assignment can be derived

by following the positions of all the zeros that are covered by the *n* lines. Otherwise follow the next step is followed.

3. Within the reduced cost matrix, we find out the smallest non zero element which is not covered by the *cut-lines* and subtract it from all the elements of the cost matrix that are not covered by the *cut-lines.* Further this value is added to all the elements which are covered by the *cut-lines*. Go back to step-2.

The overall complexity of the algorithm is $O(n^3)$ [Stachniss 2006]. The most expensive part of the algorithm is computing the number of horizontal and vertical lines covering all zeros in the cost matrix (i.e. Step-2). However, it is shown to work well with the team size of more than hundred robots and similar number of tasks [Gerkey 2004a]. In the proposed approach we have used this algorithm for assigning *frontier clusters* to the individual robots in the multi-robot system. The detailed mechanism is formally discussed in Section 4.8.1.

4.8 THE PROPOSED APPROACH

We consider the task of terrain coverage using a homogeneous team of *n* robots which can reliably communicate with a central planner. The terrain is decomposed into a grid of square cells using approximate cell decomposition method [Latombe 1991]. Each robot is equipped with sensors to detect whether or not its neighboring cells are occupied by obstacles. The robots are initialized at arbitrary locations. The multi-robot terrain coverage problem can formally be stated as follows:

Definition 1*: *Given a set R of n robots and an unknown grid like two dimensional environment E consisting of a finite set of connected free cells C, find a set of coordinated moves for each robot r* ∈ *R with initial location c* ∈ *C, so that each cell c* ∈ *C is visited by at least one robot within a finite amount of time and the overall time for coverage is minimized.

The proposed solution proceeds in the following manner: First, the central planner populates a list of frontier cells, performs *K*-means clustering on them so as to create *frontier clusters*, and allocates each *frontier cluster* thus obtained to a unique robot. The robot then computes a motion plan to pass through all frontier cells in its assigned *frontier cluster*. A pseudo code describing the algorithm is provided in Algorithm 4.1 referred to as *CAC_Planner* and Algorithm 4.2 referred to as *CAC_Robot*. A more detailed explanation of the approach follows in the subsequent sub-sections.

4.8.1 Allocation of Robots to Frontier Clusters to Robots

Let r_j be the j^{th} robot ($0 \leq j < n, j \in Z$) and its current location be p_j. Let c_i be the coordinates of the centroid of the i^{th} cluster ($0 \leq i < k, j \in Z$), where k is the parameter of *K*-means, which denotes the number of clusters desired. An efficient allocation of robots to clusters has the following desirable properties:

- The sum of the distances between the clusters and their allocated robots should be minimized. Specifically, it is the distance of a robot from the centroid of its allocated cluster.
- Optimal results should be obtained even when the number of clusters is less than the number of robots.

The cost s_{ij} of assigning r_j to the i^{th} cluster is defined as follows:

$$s_{ij} = path(c_i, p_j) \tag{4.10}$$

Here, *path(a, b)* is a function that returns the shortest path joining two points a and b, with respect to the given map (i.e., considering the locations of the known obstacles). It returns infinity if there is no path between a and b. The task allocation problem is formulated as the problem of finding an assignment matrix $A = [a_{ij}]$, where

$$a_{ij} = \begin{cases} 1 & if\ c_i\ is\ allocated\ to\ r_j \\ 0 & otherwise \end{cases} \tag{4.11}$$

such that the following expression is minimized.

$$\sum_{i=0}^{k-1} \sum_{j=0}^{n-1} a_{ij} s_{ij} \tag{4.12}$$

subject to the following constraints

$$\forall i \in [0, k), \sum_{j=0}^{n-1} a_{ij} = 1 \tag{4.13}$$

$$\forall i \in [k, n), \sum_{j=0}^{n-1} a_{ij} = 0 \tag{4.14}$$

To obtain an optimal solution for this problem in time polynomial to the number of robots, we use the Hungarian Method discussed in Section 4.7. This method takes as input a cost

matrix, describing the cost of allocating each robot to each cluster, and outputs an assignment that minimizes the total cost. The cost matrix *CM* is given by:

$$CM = [cm_{ij}] \quad \text{where,} \tag{4.15}$$

$$cm_{ij} = s_{ij} \tag{4.16}$$

There are two exceptions here; first, the Hungarian method requires that we have an equal number of tasks and workers. However, in the proposed algorithm, the number of clusters is sometimes less than the number of robots to be assigned. This case is handled by simply appending zero rows to the cost matrix and second, as the coverage task progresses towards the end, the number of frontier cells available may become lesser than the number of robots. In this case, we set the number of clusters desired equal to the total number of frontier cells available.

It makes more sense to explain this process with the help of an example. Let us consider the scenario presented in Figure 4.6. We have a set of five robots, $R = \{ r_1, r_2, r_3, r_4, r_5 \}$, and a set of five *frontier clusters*, $F = \{ f_1, f_2, f_3, f_4, f_5 \}$. Each robot has to be assigned a unique *frontier cluster* to be covered. The cost matrix is simply generated by computing the distance (i.e., the shortest path) between the centroids of the *frontier clusters* and the current positions of the individual robots as shown in Figure 4.5. In the current situation the Hungarian method finds exactly five horizontal and vertical lines that are sufficient to cover all the zeros and hence optimal assignment of robots to a unique *frontier cluster* is possible. The transformed cost matrix is shown in Figure 4.7 and the final assignment of robots to different *frontier clusters* is shown in Figure. 4.8.

	f1	***f2***	***f3***	***f4***	***f5***
R1	25	15	12	18	2
R2	13	4	1	10	9
R3	1	14	18	24	19
R4	10	6	5	12	10
R5	14	12	9	13	2

Figure 4.5 Cost/Assignment Matrix of Robots to Frontier Clusters

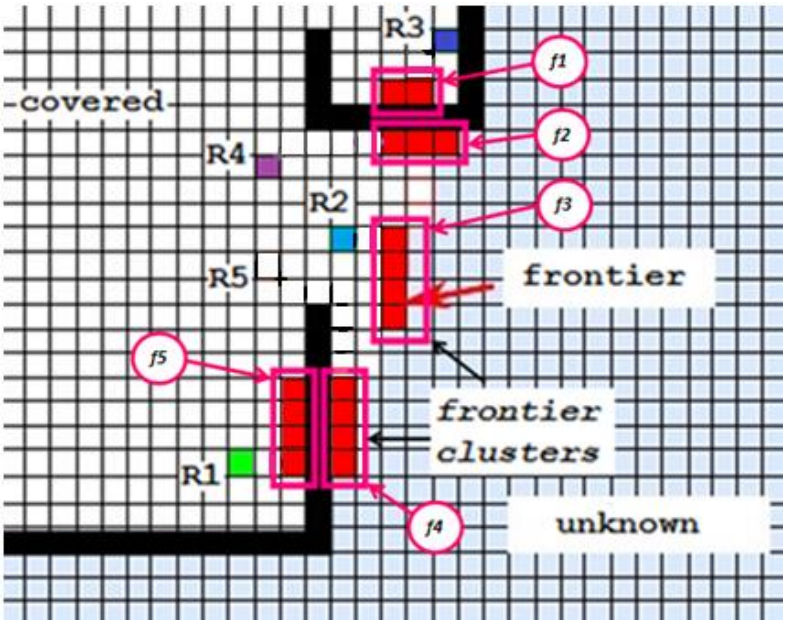


Figure 4.6 Five frontier clusters are to be allocated to five different robots

	f1	*f2*	*f3*	*f4*	*f5*
R1	19	8	6	5	**0**
R2	12	2	**0**	2	12
R3	**0**	12	17	16	22
R4	5	**0**	0	0	9
R5	8	5	3	**0**	0

Figure 4.7 Transformation of cost matrix using Hungarian method. The allocated frontier clusters are colored cells containing zeros

4.8.2 The Central Planner (or *CAC_Planner*)

The central (global) planner executes the sequence of instructions described in *Algorithm 4.1* for *CAC_Planner*. First, it populates the list of frontier cells. Subsequently, it determines the value of *K*, and performs clustering and task allocation (lines 2-14). It then waits for all the robots to transmit *task completion message* (line 16). After receiving all *task completion* messages, the global planner issues a special *endActivation* message, which indicates that task reallocation would soon occur. The robots then stop exploration and transmit their latest maps to the planner (*Algorithm 4.2*, lines 34-36, 42-44, 48). The planner then updates the global map so that its data is consistent for subsequent re-clustering and task reallocation. Each cycle of clustering, allocation, and coverage is referred to as an activation cycle. The

activation cycle is repeated until no new frontiers are discovered, upon which the planner terminates the algorithm by sending a termination message (line 20).

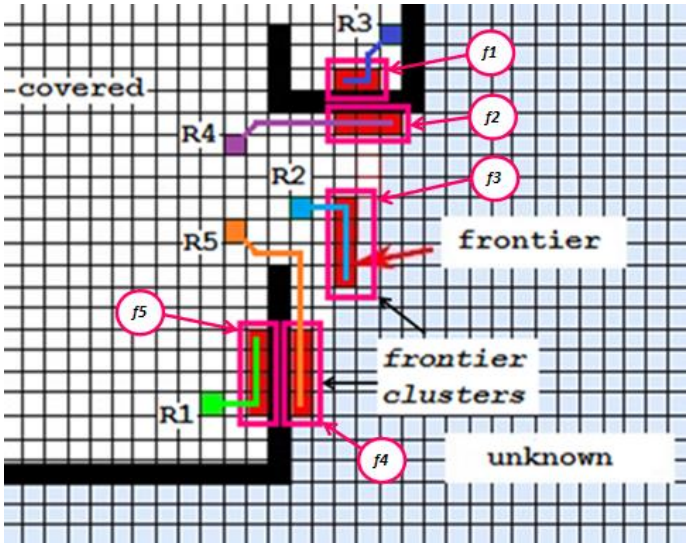


Figure 4.8 Final assignment of robots to different frontier clusters using Hungarian method

---

Algorithm 4.1 ***CAC_Planner***

1: **repeat**
2: populate frontier cell list
3: **if** *numberOfFrontierCells* < *n* **then** /* *n* = number of robots */
4: *k* ← *numberOfFrontierCells*
5: **else**
6: *k* ← *n*
7: **end if**
8: Apply *k*-means and obtain *K* clusters
9: *CM* ← **computeCostMatrix()** /* *CM* is the cost matrix */
10: **if** *K* < *n* **then**
11: Append (*n*- *k*) zero rows to *CM*
12: **end if**
13: Apply *Hungarian method* for task allocation
14: Dispatch *allocation* messages to the robots
15: Broadcast latest global map
16: Wait until all robots send *task completion* message
17: Broadcast *endActivation* message
18: Update global map
19: **until** no frontiers are visible
20: Send *termination* message to all robots

4.8.3 Local Planner (or *CAC_Robot*)

The local planner (or *CAC_Robot*) corresponds to the instructions to be followed by individual robots locally once a task is allocated to them. Algorithm 4.2 describes the method that each robot adopts to cover its allotted cluster. Robots wait for an allocation message from the planner (line 1). They then update their maps according to a global map broadcast by the planner (line 2). The allocation message contains the list of frontier cells in the allotted *frontier cluster*, referred to as *allottedList* (line 4). In addition to *allottedList*, each robot maintains two more lists - *extendedList* and *bonusList* - both of which are initially empty (lines 5-6). To reduce the distance travelled by robots, the robots continue moving to the nearest cell in the cluster from their current location. However, if this greedy strategy is adopted, the length of the path required to cover all the cells in the cluster is determined by the first cell in the cluster they move to. If this cell is not one of the end-points of the cluster, it increases the length of the path taken by the robot to cover all cells in the cluster. To prevent this drawback caused by the greedy strategy, we also consider the frontier cells adjoining the cells in the cluster. Using these additional frontier cells (referred to as extended frontier cells), robots can generate trajectories that cover the cells in the cluster with reduced redundancy. Following this strategy, the robots first move to the closest point on the allotted *frontier cluster*, from where, they reach one end of the cluster. The nearest available allotted frontier now lies in the chunk connecting the opposite end of the allotted *frontier cluster* to the cell from which coverage of this cluster started. To reach this cluster, robots can simply make use of the adjoining frontier cells discovered until this point, rather than taking the shortest path that passes through already explored territory. This strategy was incorporated in the proposed approach in the manner shown in Algorithm 4.2 (lines 7-26). The algorithm exploits the inherent order in which individual grid cells are stored in the robot's (or planner's) memory. In all nearest frontier choices involved (lines 8, 9), frontier cells with the same distance are ordered by their coordinates (either row-major or column-major order could be used). Moreover, when the nearest cells exist in both *allottedList* and *extendedList*, a cell from *allottedList* is preferred (lines 19-23).

Algorithm 4.2 ***CAC_Robot***

1: Wait for *allocation* message from planner
2: Receive latest global map from planner
3: **while** termination message not received **do**
4: *allottedList* ← list of cells in allotted cluster

```
5:          extendedList ← empty list
6:          bonusList ← empty list
7:          while there is an unvisited cell in allottedList or in extendedList do
8:              C_a ← first nearest cell in allottedList
9:              C_e ← first nearest cell in extendedList
10:             if C_e is nearer than C_a then
11:                 chosenCell ← C_e
12:             else
13:                 chosenCell ← C_a
14:             end if
15:             while chosenCell is not visited do
16:                 make a move towards chosenCell
17:                 F_new ← list of frontiers newly discovered
18:                 for f ∈ F_new do
19:                     if f is a neighbor of any c (cell) ∈ allottedList then
20:                         append(extendedList, f)
21:                     else
22:                         append(bonusList, f)
23:                     end if
24:                 end for
25:             end while
26:         end while
27:         Dispatch completion message to planner
28:         while endActivation message not received do
29:             if there is an unvisited cell in bonusList then
30:                 chosenCell ← first nearest cell in bonusList
31:                 while chosenCell is not visited do
32:                     Make a move towards chosenCell
33:                     append(bonusList, newly discovered frontiers)
34:                     if endActivation message received then
35:                         break
36:                     end if
37:                 end while
38:             else
39:                 chosenCell ← first nearest frontier cell visible
40:                 while chosenCell is not visited do
41:                     Make a move towards chosenCell
42:                     if endActivation message received then
43:                         break
44:                     end if
45:                 end while
46:             end if
47:         end while
48:         Send latest map to the central planner
49:     end while
```

The manner in which the proposed path planning strategy works on each of the robots is illustrated in Figure 4.9 and Figure 4.10. Here, the cells in blue indicate the *allottedList* and the cells in green indicate the *extendedList*. Cells in black are obstacles. The robot is currently located in a cell which is colored yellow. After all frontier cells in *allottedList* and *extendedList* are visited, the robot dispatches a completion message to the planner, indicating that it has completed its share of work. When the planner receives completion messages from all robots, it performs map updates and begins the next iteration. Until then, each robot that dispatched a completion message visits the remaining frontier cells it discovered, which are stored in *bonusList*. During this process, if any new frontier cells are discovered, they are added to *bonusList*, and subsequently covered (lines 29-37). If a robot runs out of frontier cells in *bonusList* and still does not receive an allocation message, it moves towards the first nearest frontier it can find (lines 38-46). This ensures that, upon subsequent reception of an allocation message, the robot is close to new areas being explored. This also ensures that the resources (here robots) of the multi-robot system are utilized effectively.

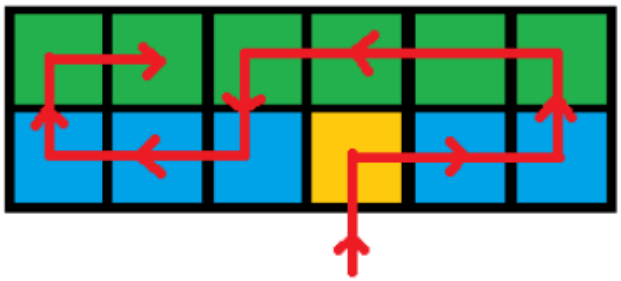


Figure 4.9 The proposed Motion Planning Strategy (without obstacles)

Figure 4.10 The Proposed Motion Planning Strategy (with obstacles)

4.9 Experimental Results And Analysis

The three state of the art representative approaches discussed in Section 4.3 and the proposed approach have been implemented and extensively evaluated both in simulation and in a controlled laboratory setting on a team of four Firebird V [Firebird 2016] robots. The proposed approach is compared with the other three approaches by varying the number of

robots, and the complexity of the map. In Section 4.9.1, the architecture of the multi-robot system is described. The quantitative results obtained from the simulation and from the actual experiments are discussed in Section 4.9.2 and Section 4.9.3 respectively.

4.9.1 System Architecture

The architecture of the system is shown in Figure 4.11. The central planner and robots are composed of multiple layers, with each layer responsible for the execution of a well-defined set of tasks. Both the central planner and the robots have two layers in common i.e., the coordination layer and the planning layer. The *coordination layer* is responsible for facilitating communication between the central planner and the robots. From the central planner the following messages are sent to the robots:

- *Task allocation message* containing the set of frontiers to be covered.
- *Global map updates* are sent immediately after the task assignment.
- *End activation message* indicating the start of the re-clustering operation.
- *Termination message* indicating the end of the coverage task.

The robots send the following messages to the central planner:

- *Task completion message* indicating that the assigned task *T* is complete. The robot begins the coverage of those frontier cells which it has discovered on its own accord while pursuing the previously assigned task *T*.
- *Local map updates* are sent only when the end activation message is received from the central planner.

Whenever there is a communication received from either side the message is serviced by the respective entity as soon as it finishes the operation it is currently executing.

For the *central planner* the *planning layer* is responsible for various high level tasks. Its primary responsibilities are:

- *Path planning* - Jump point search algorithm [Harabor 2011] is used for this purpose.
- *Geodesic K-means clustering* - clusters the frontier cells and creates frontier clusters based on distances obtained from the path planner.
- *Task Allocation* - executes the Hungarian method for finding optimal assignment of robots to frontier clusters.

- *Map maintenance* - updates the global map after receiving the completion message from all the robots.

The *planning layer of the robot* is responsible for the following tasks:

- *Map maintenance* - updates the local instance of the global map after receiving map updates from the global planner and after sensing its neighborhood using its range sensors i.e., the eight connected cells surrounding the robot's current position.
- *Strategy selection* - when the robot finishes the assigned task before receiving the end activation message from the planner (which means the *allottedList* and the *extendedList* are empty) the strategy selector intimates the motion planner, which then begins to generate trajectories from robots to the cells in the *bonusList*.

The *low-level controller layer* is responsible for generating control signals. In the presented architecture, each layer communicates directly only with the layer immediately preceding or succeeding it. Arrows in the figure indicate the direction of communication among various modules.

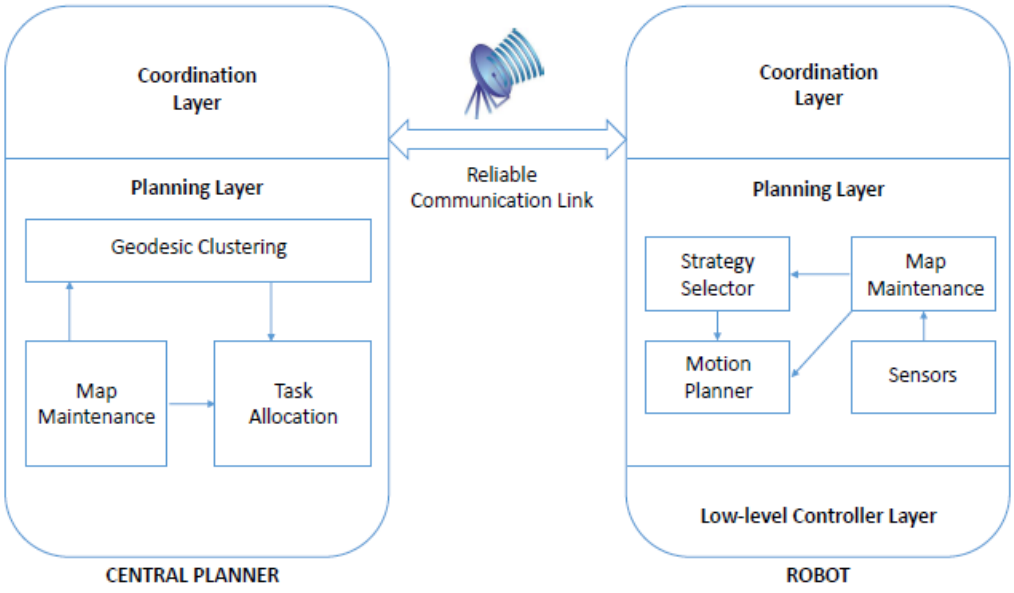
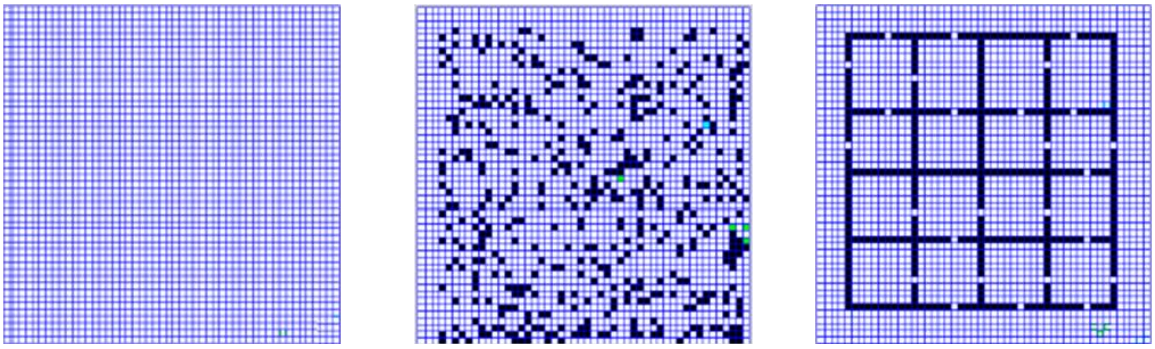


Figure 4.11 Architecture of the Multi-Robot System

4.9.2 Simulation Results

Terrains of various dimensions and obstacle distributions are used in simulation. The completion time and redundancy in coverage is recorded. The proposed approach is compared with [Burgard 2005], [Yamauchi 1998], and [Puig 2011]. Figure 4.12 shows three different simulated terrain maps used for terrain coverage. The walls and obstacles are

represented by black areas and the free space that needs to be covered is represented by the white areas. Figure 4.12(a) is a completely free region devoid of any walls and obstacles. Figure 4.12(b) is a representation of some outdoor cluttered environment with lots of obstacles lying here and there. The map in Figure 4.12(c) is a sort of an office map with door openings in connected rooms and long walls. In all the maps it is assumed that the free space to be covered is a one single connected component. The set of robots R = {2, 4, 6, 8, 10} are considered for the simulation purpose in each map. In each map a minimum of 25 simulation runs is conducted by increasing the number of robots from 2 to 10. In each simulation run the robots are made to start from random initial positions.



(a) Free World (b) Outdoor Map (c) Office Map

Figure 4.12 Three different terrains used for simulation (dimensions: 50 x 50 cells)

The behavior of the proposed approach is assessed in a quantitative manner on the basic of the results obtained from computer simulation. The discussion presented below is based on the completion time and percentage redundancy graphs of different approaches considered for evaluation and are shown in Figure 4.13 to 4.15 and Figure 4.16 to 4.18 respectively.

In [Burgard 2005], the robots are dispersed locally because the movement of robots is restricted in a manner that they maintain non-overlapping sensing impressions. The robots tend to choose targets that are in close vicinity of each other. The robots start from random initial positions. After a while they come close to each other and do not spread out further as a result overlap in coverage and coverage completion time increases. Increasing the number of robots does reduce the time to complete coverage but the percentage redundancy increases drastically.

In [Puig 2011], global dispersion is targeted and the robots are sent to the targets far from each other in the unknown region. Since the map of the terrain is not known beforehand and forward planning is not possible therefore Euclidean distances are used. The robots leave behind many uncovered cells that were discovered while they were getting dispersed and

return to cover these frontier cells later. The robots have to plan their paths through the already covered cells on several occasions and the approach is not able to take the advantage to global dispersion. The task allocation mechanism used in this approach is not able to exploit the information generated by the robots in real-time. As a result, coverage completion time is increased. After re-planning is introduced in this approach it gives better performance than [Burgard 2005].

In [Yamauchi 1998], the authors have suggested a purely greedy approach i.e. the robots select the nearest frontier cell for coverage. The robots do not coordinate with their peers when selecting a frontier cell for coverage. The approach does not confine the robots in a limited sensing range. This results in some robots choosing the same frontier cell for the coverage if they are operating in a close vicinity. This results in increased overlapping coverage. It is shown in a comparative study of different exploration algorithms [Juliá 2012] that [Yamauchi 1998] gives better performance than [Burgard 2005]. Also [Yamauchi 1998] performs better than the [Puig 2011] because the robots are greedy and do not run very far from the frontier cells discovered in the recent past.

The proposed approach (CAC) completes coverage faster than the other approaches in all the maps and by increasing the number of robots. This is because of the path planning scheme that results in shorter paths for coverage. Furthermore, the Hungarian method ensures an optimal assignment of robots to clusters. The robots do not remain idle after completing their assigned task, therefore, increasing the overall utilization of the system resources (robots).

Following are three important observations based on the simulation results:

(a) It is evident from the Figure 4.13, 4.15 and 4.17 that, irrespective of the nature of the map and the terrain coverage approach considered for evaluation, the team of mobile robots performs better in terms of completion time as the number of robots are increased.

(b) It is evident from the Figure 4.14, 4.16 and 4.18 that, irrespective of the map and the terrain coverage approach considered for evaluation, increasing the number of robots results in increased redundant coverage. Redundancy is estimated by counting the number of steps that the robots take which do not result in a new cell being covered.

(c) The percentage improvement of the proposed approach over other representative approaches in terms of coverage completion time and redundant coverage are presented in Table 4.1, 4.2, and 4.3 for the Free World, Outdoor and Office maps respectively. It

can be observed that, irrespective of the terrain map and the robot team size, proposed approach always performs better than the other state-of-the-art approaches.

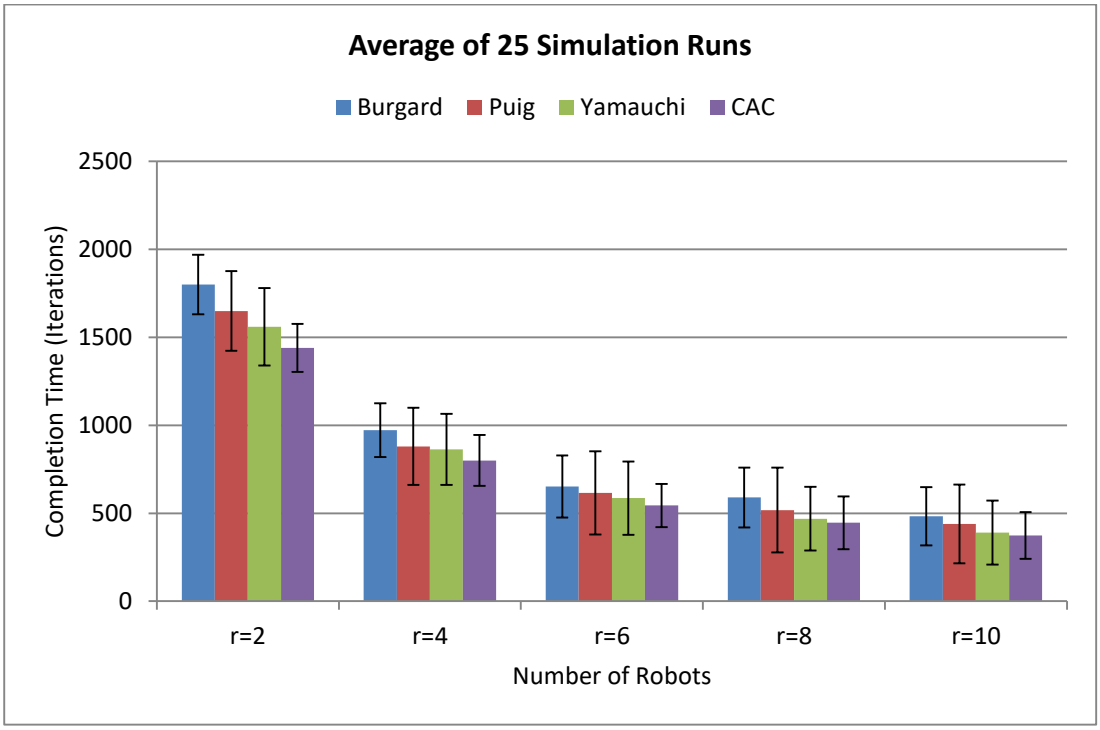


Figure 4.13 Completion time measured in the Free World map

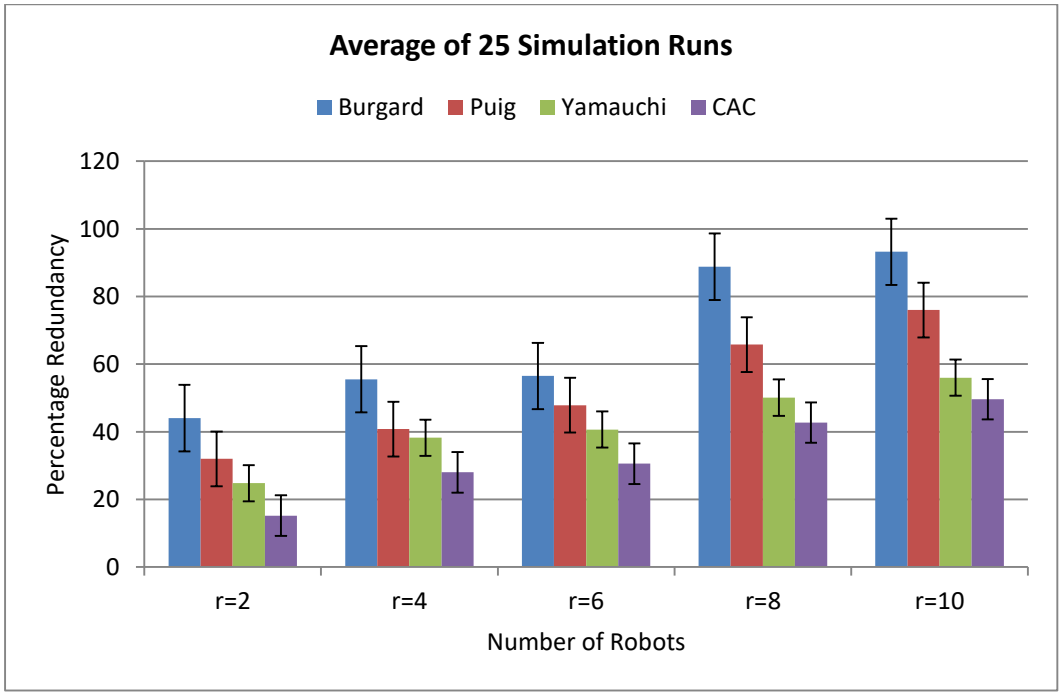


Figure 4.14 Percentage Redundancy measured in the Free World Map

Table 4.1 Comparison of CAC with different approaches in Free World map

MAP	ROBOTS	CAC vs.	PERCENTAGE IMPROVEMENT IN COVERAGE COMPLETION TIME	PERCENTAGE IMPROVEMENT IN NON-OVERLAPPING COVERAGE
FREE WORLD MAP	2	Burgard	20	65.45
		Puig	12.73	52.5
		Yamauchi	7.69	38.71
	4	Burgard	17.7	49.57
		Puig	9.09	31.37
		Yamauchi	7.41	26.78
	6	Burgard	16.56	45.89
		Puig	11.69	36.12
		Yamauchi	7.17	24.8
	8	Burgard	24.41	51.89
		Puig	13.9	35.04
		Yamauchi	4.9	14.7
	10	Burgard	22.57	46.78
		Puig	15	34.74
		Yamauchi	4.1	11.43

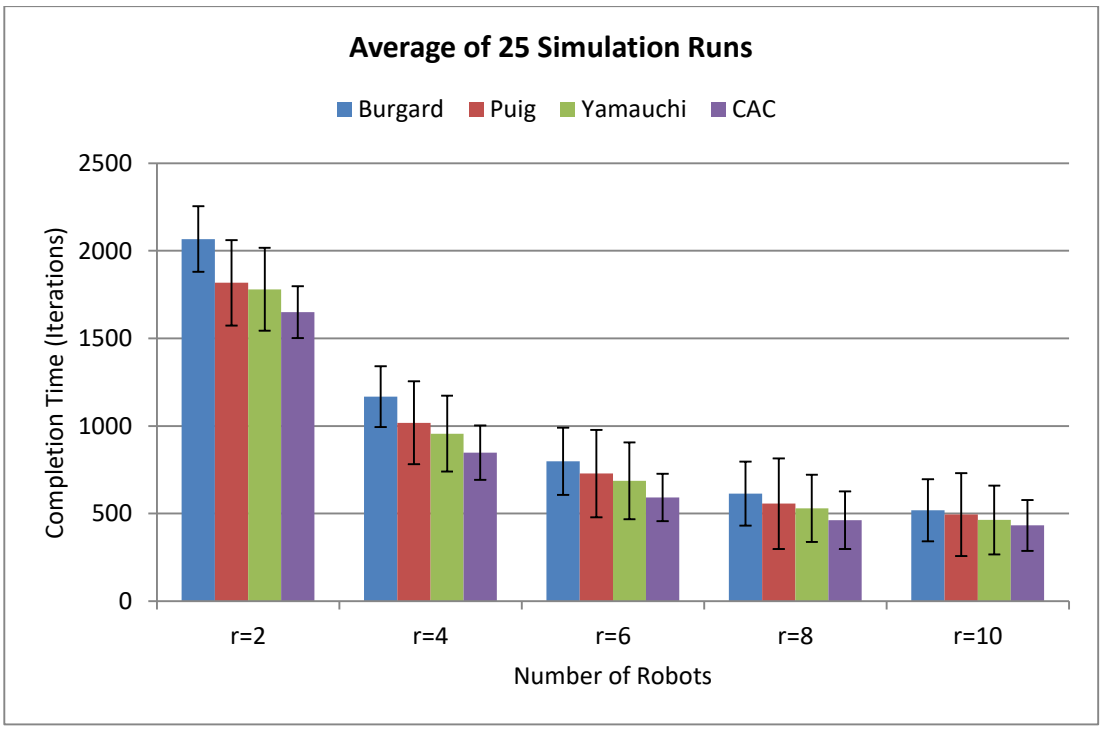


Figure 4.15 Completion time measured in the Outdoor map

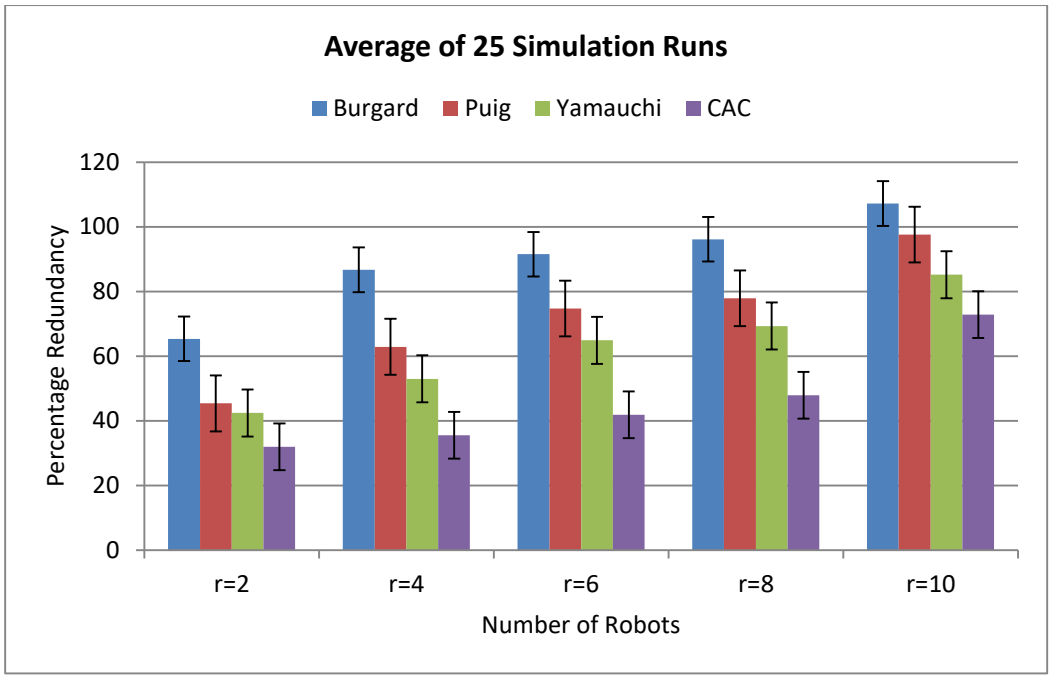


Figure 4.16 Percentage Redundancy measured in the Outdoor Map

Table 4.2 Comparison of CAC with different approaches in Outdoor map

MAP	ROBOTS	CAC vs.	PERCENTAGE IMPROVEMENT IN COVERAGE COMPLETION TIME	PERCENTAGE IMPROVEMENT IN NON-OVERLAPPING COVERAGE
OUTDOOR MAP	2	Burgard	20.22	51.16
		Puig	9.25	29.63
		Yamauchi	7.36	24.72
	4	Burgard	27.42	59.04
		Puig	16.8	43.51
		Yamauchi	11.4	32.93
	6	Burgard	25.94	54.28
		Puig	18.82	44
		Yamauchi	13.97	35.51
	8	Burgard	24.63	50.25
		Puig	16.91	38.6
		Yamauchi	12.67	30.95
	10	Burgard	16.6	32.09
		Puig	12.55	25.41
		Yamauchi	6.7	14.55

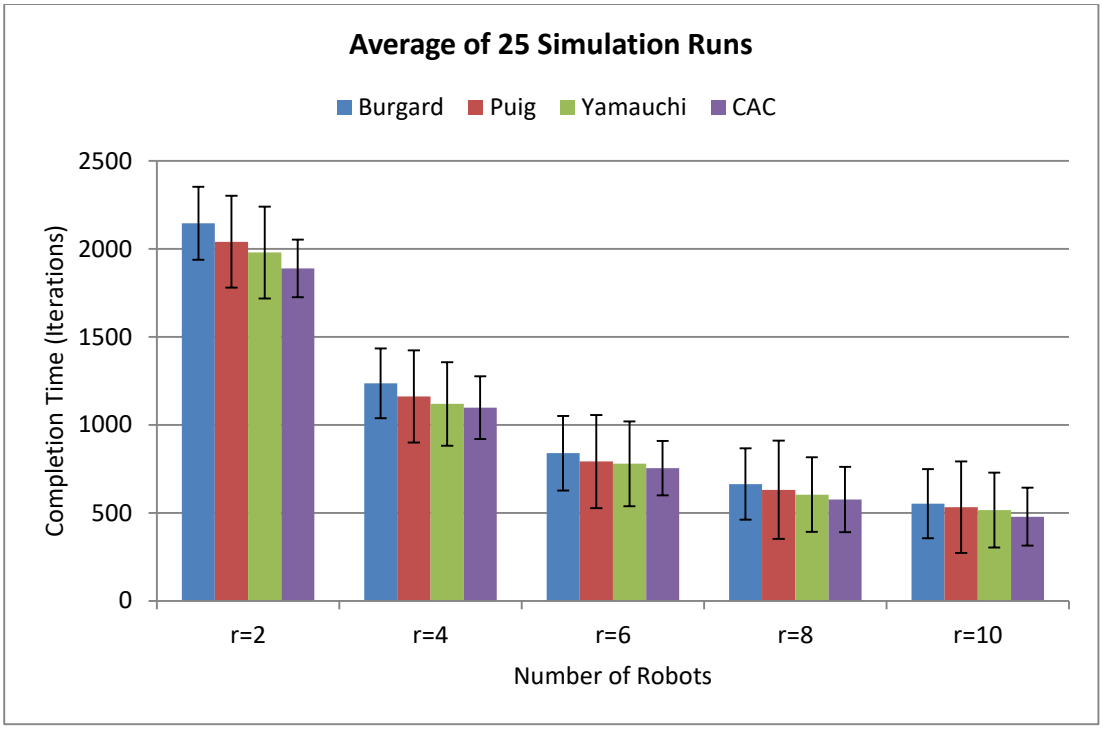


Figure 4.17 Completion time measured in the Office map

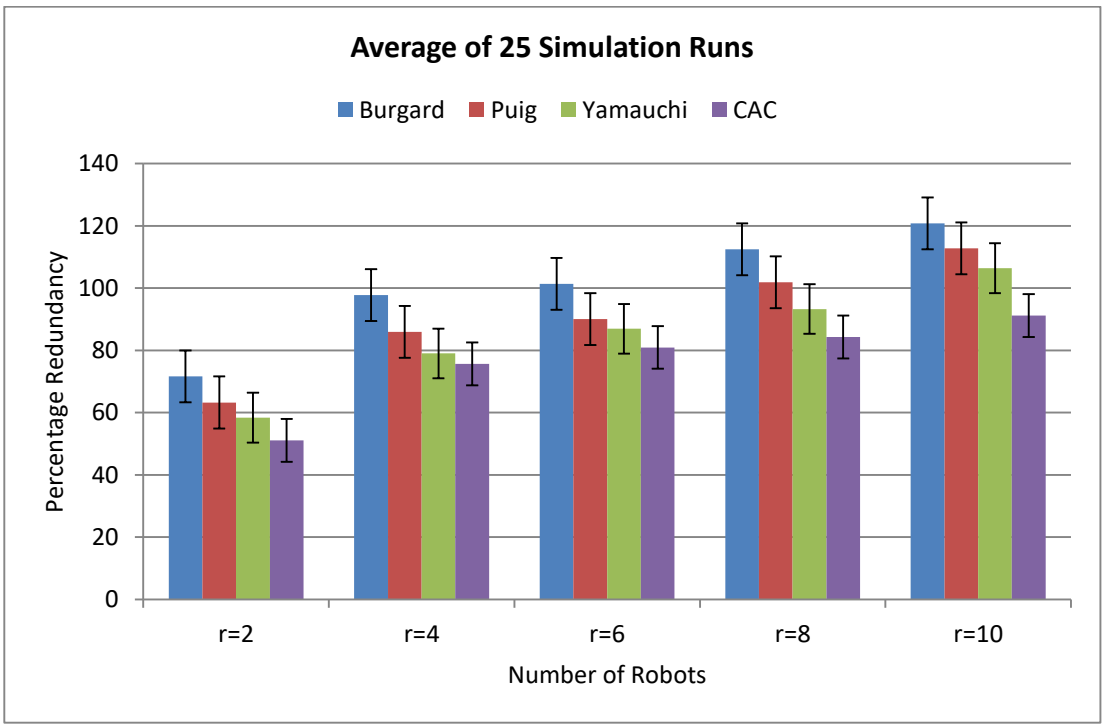


Figure 4.18 Percentage Redundancy measured in the Office Map

Table 4.3 Comparison of CAC with different approaches in Office map

MAP	ROBOTS	CAC vs.	PERCENTAGE IMPROVEMENT IN COVERAGE COMPLETION TIME	PERCENTAGE IMPROVEMENT IN NON-OVERLAPPING COVERAGE
OFFICCE MAP	2	Burgard	11.98	28.68
		Puig	7.45	19.22
		Yamauchi	4.6	12.47
	4	Burgard	11.17	22.59
		Puig	5.51	11.92
		Yamauchi	1.88	4.25
	6	Burgard	10.13	20.13
		Puig	4.8	10.12
		Yamauchi	3.21	6.9
	8	Burgard	13.25	25.04
		Puig	8.72	17.27
		Yamauchi	4.64	9.61
	10	Burgard	13.41	24.5
		Puig	10.15	19.15
		Yamauchi	7.36	14.29

Dispersion of robots across the map is not an explicitly expressed goal of the proposed approach. Rather, it is an emergent behavior. This inference is evident from the graph of Figure 4.19. This plot shows the sum of variances of cluster centroids as exploration progresses in a free map, using nine robots that start as a cluster at the center. It can be understood that, although the degree of dispersion is initially local (confined to a particular region of the map), as the coverage task progresses, it becomes global (spread across larger portions of the map). Figure 4.20 shows the volume of cells to be clustered in each activation cycle by the proposed approach when compared with [Puig 2011]. The proposed approach spends very less time doing clustering because the number of cells to be clustered are very few.

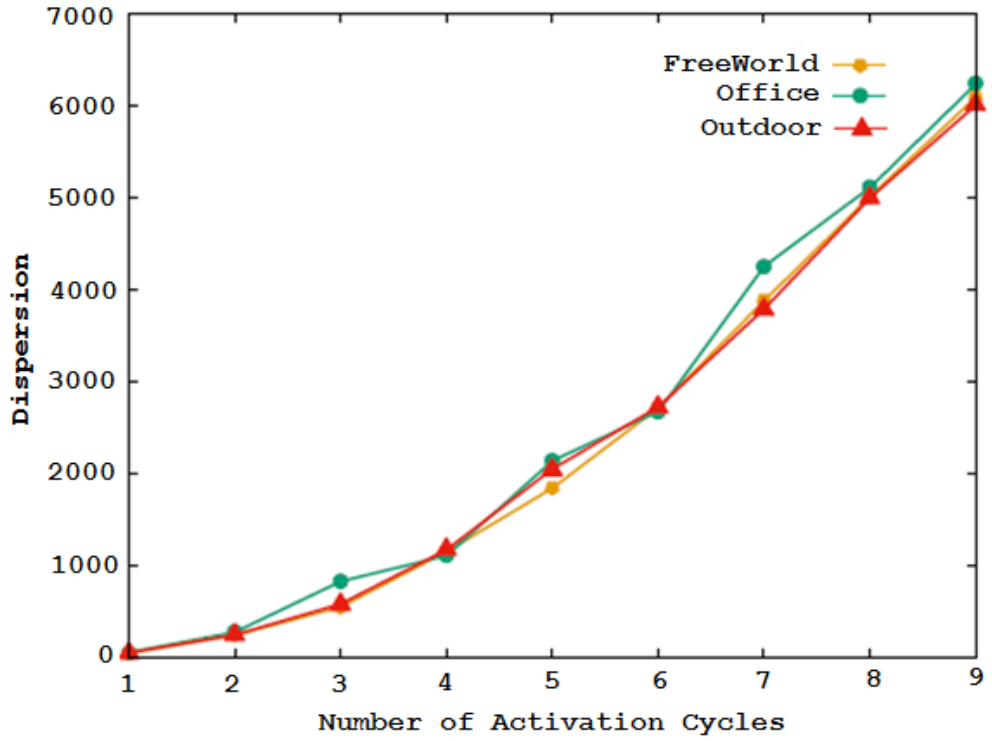


Figure 4.19 Nature of dispersion (Free World, Office, and Outdoor maps)

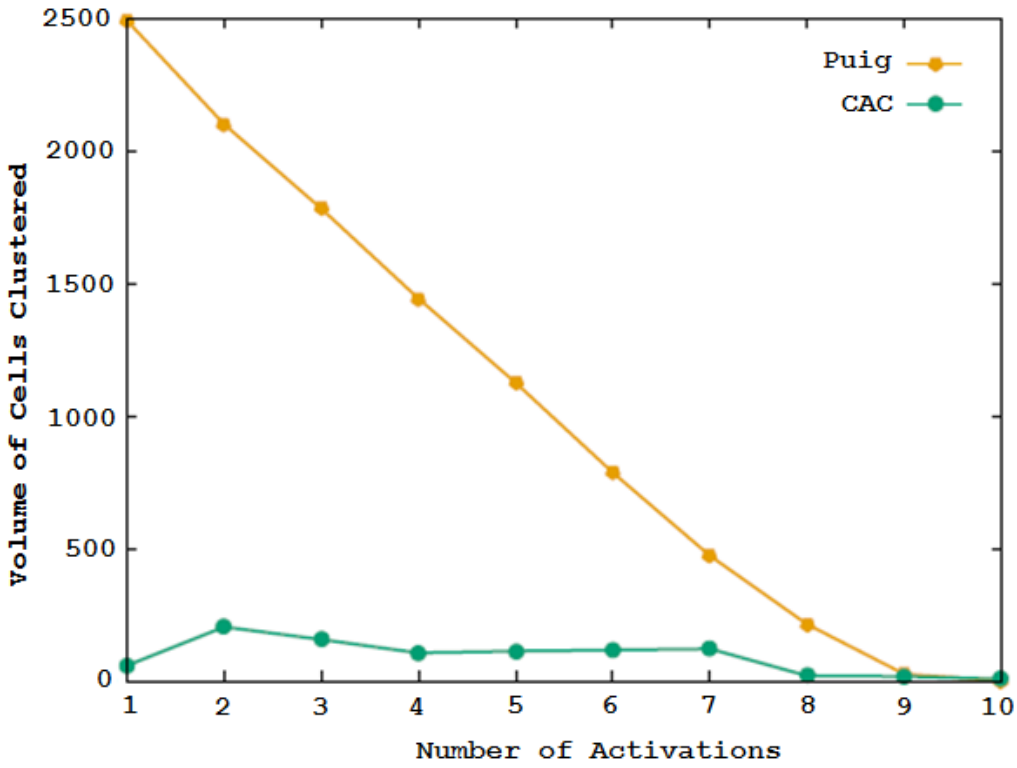
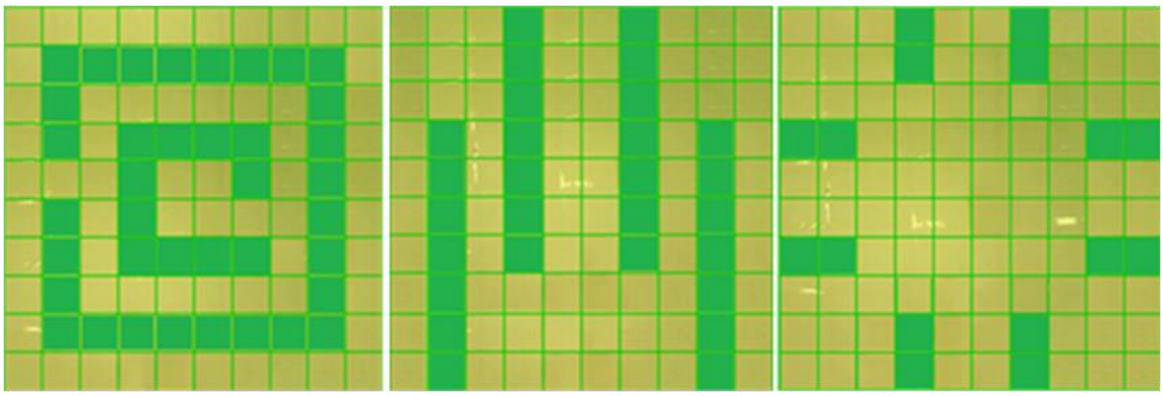


Figure 4.20 Volume of Cells Clustered (Free World map, 8 Robots)

4.9.3 Experimental Results

To demonstrate the feasibility of the proposed approach, it was also implemented on a team of Firebird V robots [Firebird 2016]. Each robot in the team has an AVR ATMega2560 microcontroller. 2.4 GHz XBee modules were used for communication among robots and the central planner. These modules allowed for communication without interference on 16 different channels. For localization purposes, a visual marker based positioning system was deployed. This positioning system was used to communicate each robot's location to the central planner and to the robot itself. The robots were equipped with infrared-based proximity sensors in eight directions to detect obstacles in each neighboring cell. These sensors could reliably detect obstacles within a distance of 40 cm. Figure 4.21 shows three different settings for the purpose of comparison of different approaches considered for evaluation. The maps are discretized into 10×10 square grid cells of size equal to the footprint of the robot which is 0.2 m^2. The actual time taken by the robots to complete the coverage task (in seconds) is shown in the graphs of Figure 4.22 to 4.24. The proposed approach (CAC) is compared with [Yamauchi 1998] and [Burgard 2005]. [Puig 2011] is not evaluated in experiments because it targets global dispersion, the effect of which is visible only in bigger maps.



(a) Circular Maze Map (b) Bar Maze Map (c) Living Room Map

Figure 4.21 Three different maps used in terrain coverage experiment

The completion time trends and percentage redundancy trends observed in experiments are no different than what we have seen in the simulations. Following are the observations that are made regarding the completion time from Figures 4.22, 4.24, and 4.26 and percentage redundancy graphs shown in Figures 4.23, 4.25, and 4.27.

It can be inferred from Figure 4.22 that the proposed approach (CAC) performs better than [Burgard 2005] and [Yamauchi 1998] in terms of completion time for a Circular Maze Map. This is due to the complexity of the Circular Maze map wherein it is observed that for [Burgard 2005] and [Yamauchi 1998] algorithms the robots just follow each other while trying to reach their chosen frontier cells. Also, as seen in Figure 4.23, in the Circular Maze map the percentage redundancy for [Burgard 2005] and [Yamauchi 1998] algorithms are higher than that for CAC. The percentage improvement of CAC over [Burgard 2005] and [Yamauchi 1998], both in terms of completion time and non-overlapping coverage is shown Table 4.4.

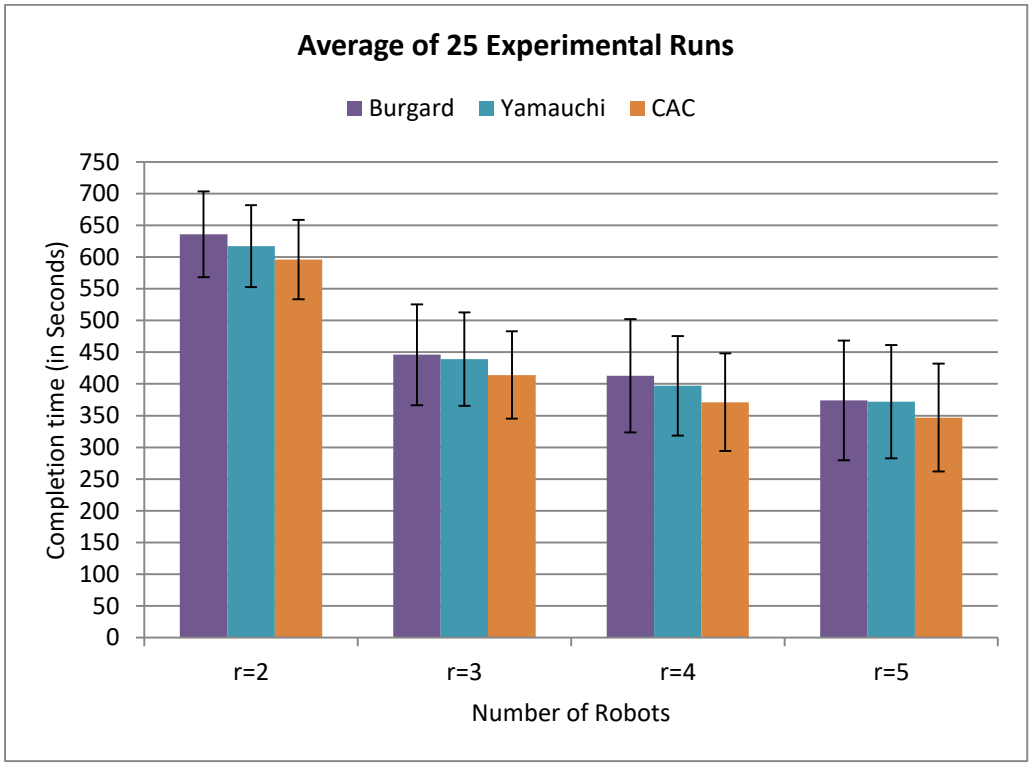


Figure 4.22 Completion time measured in the Circular Maze map

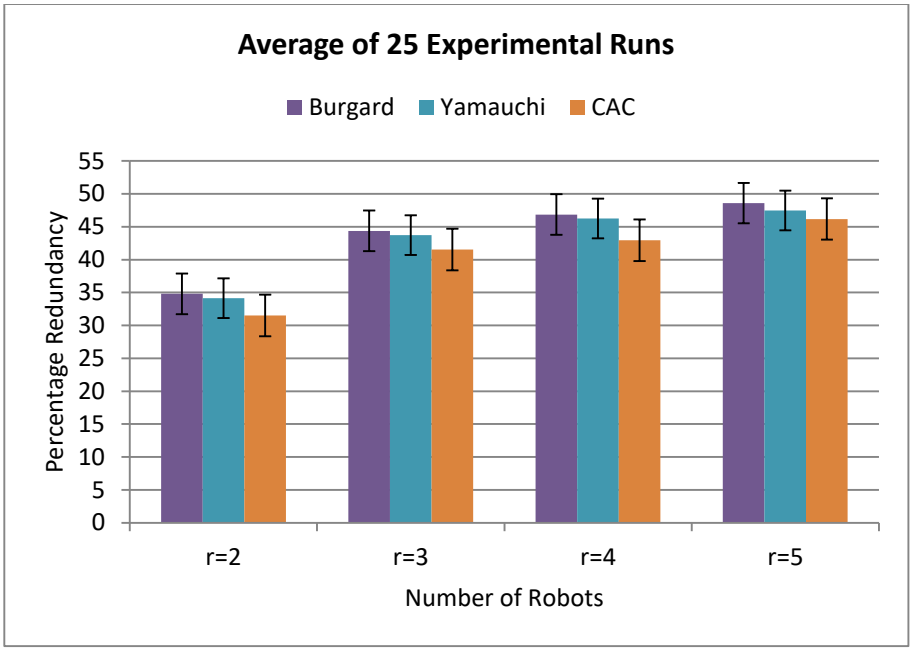


Figure 4.23 Percentage Redundancy measured in the Circular Maze map

Table 4.4 Comparison of CAC with different approaches in Circular Maze map

MAP	ROBOTS	CAC vs.	PERCENTAGE IMPROVEMENT IN COVERAGE COMPLETION TIME	PERCENTAGE IMPROVEMENT IN NON-OVERLAPPING COVERAGE
CIRCULAR MAZE	2	Burgard	6.29	9.41
		Yamauchi	3.4	7.64
	3	Burgard	7.17	6.38
		Yamauchi	5.69	5.04
	4	Burgard	10.17	8.37
		Yamauchi	6.55	7.14
	5	Burgard	7.22	4.97
		Yamauchi	6.72	2.68

The completion time graphs of Bar Maze map and Living Room map are shown in Figure 4.24 and Figure 4.26 respectively. It is inferred that the performance of CAC has improved for less complex maps since there are more number of unobstructed frontier cells to cover. As the number of robots are increased all approaches starts performing better in their own comparison. The percentage redundancy in Bar Maze map and Living Room map is shown in Figure 4.25 and Figure 4.27. It can be observed that coverage achieved by CAC is the least redundant in these two maps. As a general trend it can be noticed that irrespective of the number of robots used, the percentage redundancy for all the approaches decreases as the

complexity of the maps is reduced. On the other hand, it is also observed that irrespective of the map used, the percentage redundancy increases with the increase in the number of robots. The percentage improvement of CAC over [Burgard 2005] and [Yamauchi 1998], both in terms of completion time and non-overlapping coverage is shown Table 4.5 and Table 4.6 for Bar Maze and Living Room maps respectively.

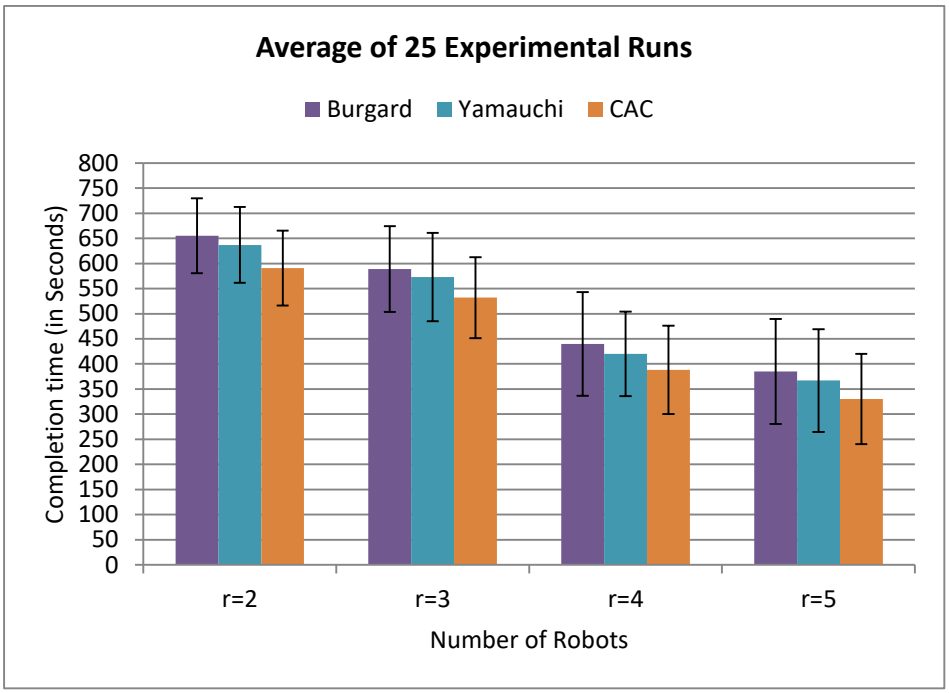


Figure 4.24 Completion time measured in the Bar Maze map

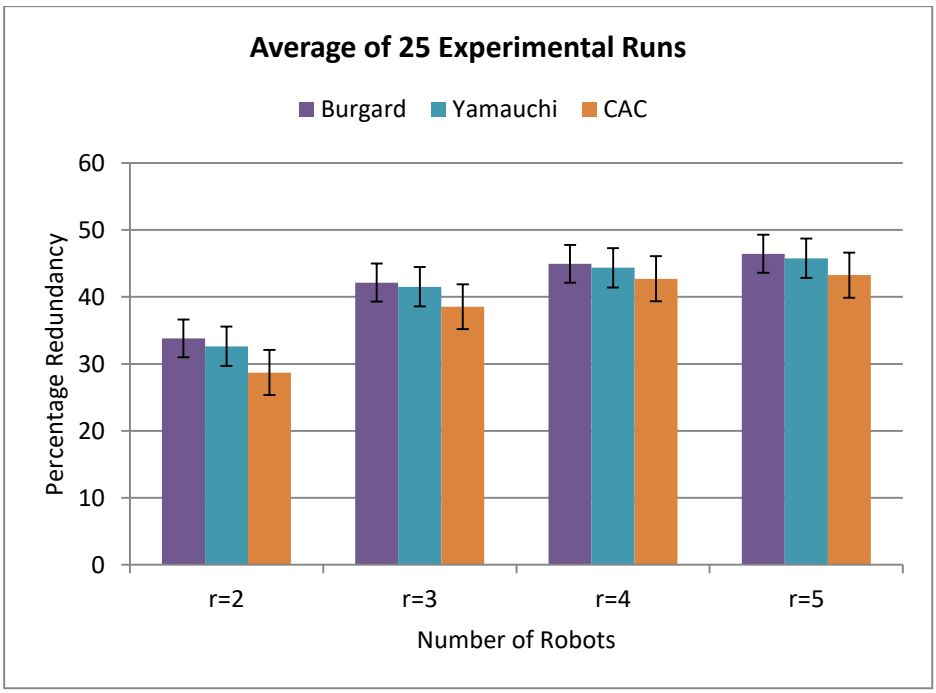


Figure 4.25 Percentage Redundancy measured in the Bar Maze map

Table 4.5 Comparison of CAC with different approaches in Bar Maze Map

MAP	ROBOTS	CAC vs.	PERCENTAGE IMPROVEMENT IN COVERAGE COMPLETION TIME	PERCENTAGE IMPROVEMENT IN NON-OVERLAPPING COVERAGE
BAR MAZE	2	Burgard	9.77	15.05
		Yamauchi	7.22	12
	3	Burgard	9.68	8.56
		Yamauchi	7.16	7.19
	4	Burgard	11.82	5.01
		Yamauchi	7.62	3.71
	5	Burgard	14.29	6.87
		Yamauchi	10.08	5.5

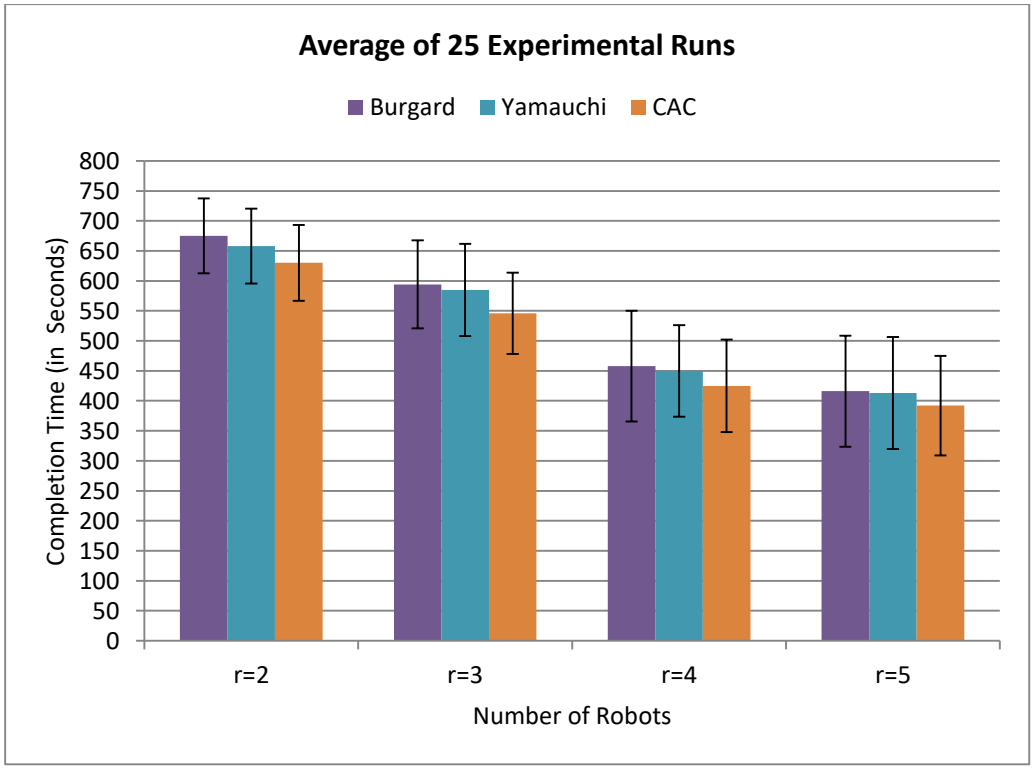


Figure 4.26 Completion time measured in the Living Room map

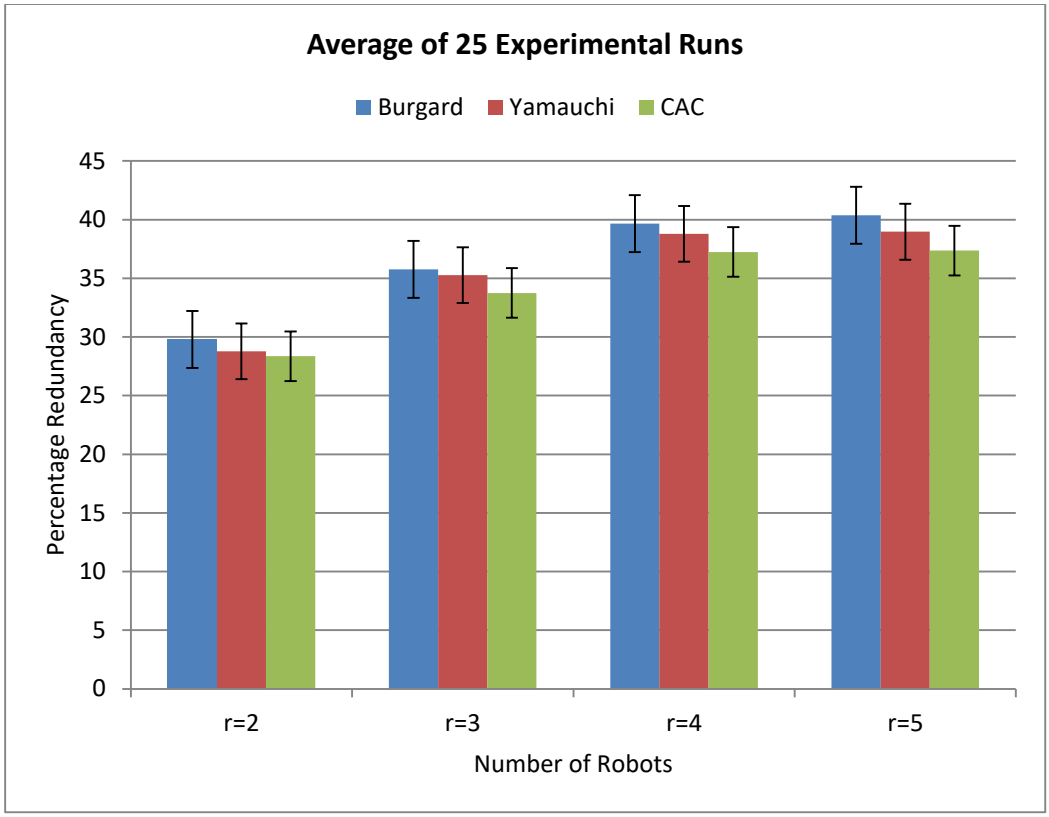


Figure 4.27 Percentage Redundancy measured in the Living Room map

Table 4.6 Comparison of CAC with different approaches in Living Room Map

MAP	ROBOTS	CAC vs.	PERCENTAGE IMPROVEMENT IN COVERAGE COMPLETION TIME	PERCENTAGE IMPROVEMENT IN NON-OVERLAPPING COVERAGE
LIVING ROOM	2	Burgard	6.67	4.76
		Yamauchi	4.26	1.45
	3	Burgard	8.08	5.61
		Yamauchi	6.67	4.33
	4	Burgard	7.21	6.1
		Yamauchi	5.56	3.99
	5	Burgard	5.77	7.47
		Yamauchi	5.08	4.15

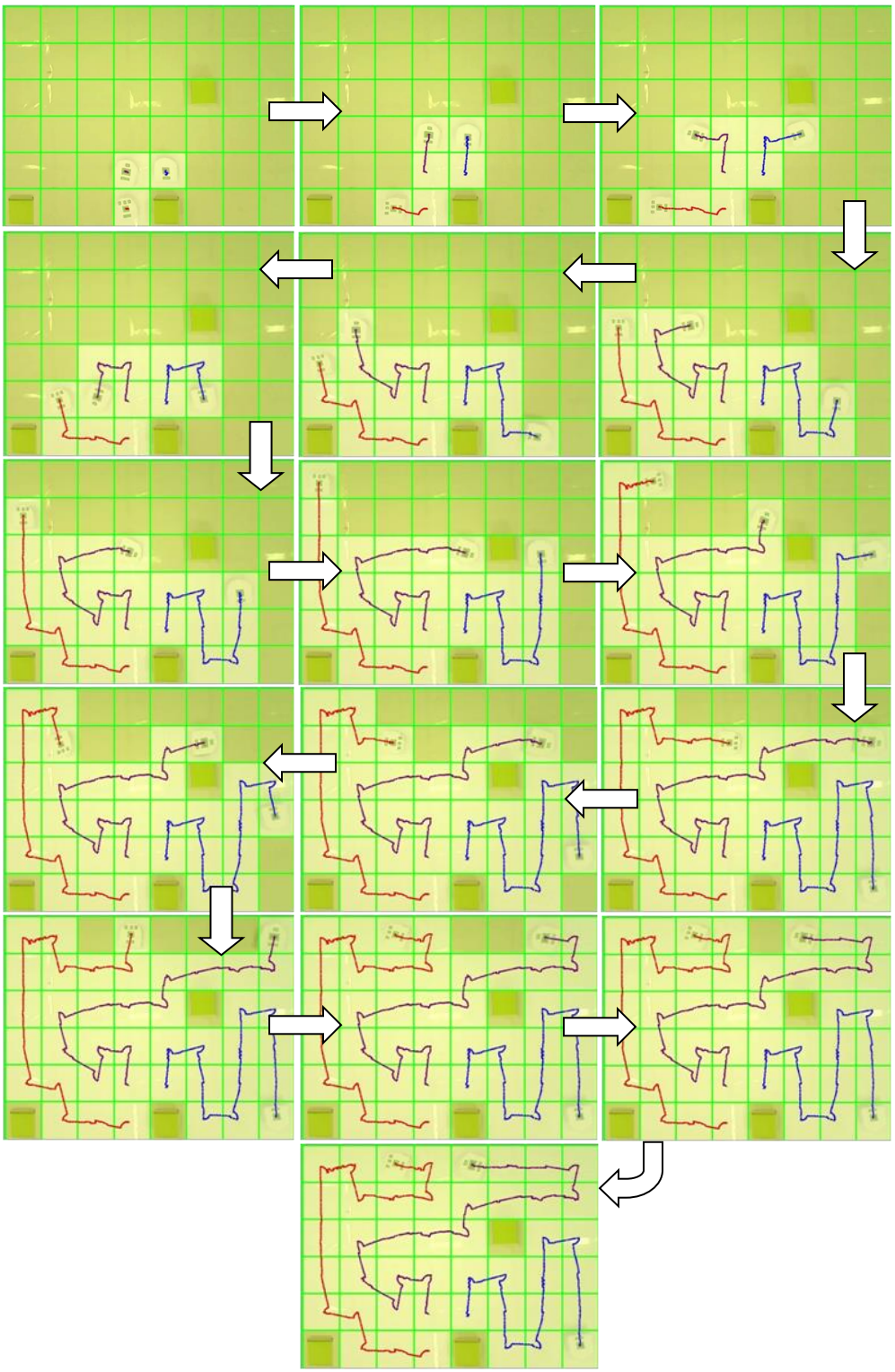


Figure 4.28 Three Firebird V Robots Executing the CAC Algorithm

4.10 CHAPTER SUMMARY

In this chapter the problem of online terrain coverage using multiple mobile robots is considered. Terrain coverage requires the robots to physically traverse the entire terrain i.e., whole of the un-obstructed and connected region. The approach proposed in this chapter is a centralized approach, comprising of a central planner that is responsible for multi-robot coordination. Preparation of the task subsets, optimal task allocation, map maintenance and re-planning are some of the vital functionalities of the central planner. On the other hand, individual robots sense their surroundings and maintain their own local map with respect to the task subset allocated to them. The robots periodically update the central planner and carry out motion planning. Approximate cell decomposition has been used as a terrain decomposition method. Three Firebird V robots are used for experiments.

It is assumed that the terrain is bounded, however, neither the map of the terrain nor the bounds of the terrain are known a priori. The location of each robot with respect to some global coordinate system is known to the central planner and to the robots themselves. The robots can communicate with the central planner at all times. The suggested algorithm proceeds with the minimal knowledge of the already explored region and the frontier cells. It creates clusters of frontier cells (task subsets) which are apportioned to the robots using an optimal assignment scheme. Coverage is then performed using a path planning technique that results in lesser overlapping coverage. The proposed algorithm performs better than some of the state of the art approaches that are representative of multi-robot terrain coverage and exploration. The main motivation behind this research comes from the fact that many frontier propagation algorithms like [Burgard 2005] and [Yamauchi 1998] are more exploitatory in coverage and highly conservative in exploring new regions. These algorithms thus result in local dispersion. On the other hand, approaches which partition the environment into bigger regions i.e., [Puig 2011] are more exploratory and in search of new regions for coverage leave behind uncovered regions. The approach proposed in this chapter completely exploits its knowledge of the frontier cells and does not leave behind any uncovered cells. Dispersion is not explicitly targeted rather; it is an emergent behavior. As the map unfolds the robots starts exploring faster. The improvement obtained by the proposed approach over the state-of-the-art approaches is as a result of proper coordination between the central planner and the multi-robot team. The central planner conducts global planning and allows the individual robots to completely execute the task assigned to them. The individual robots execute their local plan and at the same time optimize upon their

motion plan such that the overlapping coverage is minimized. However, the proposed approach does not handle failures and is not robust. In the next chapter we present a completely decentralized fault tolerant algorithm for online terrain coverage which follows a structured trajectory approach.

CHAPTER 5

SYNCHRONOUS FRONTIER ALLOCATION FOR SCALABLE ONLINE MULTI-ROBOT TERRAIN COVERAGE

5.1 INTRODUCTION

Online terrain coverage is a problem which has gathered substantial attention from the multi-robot systems research community. The nature and complexity of the problem was explained in Chapter 4. It is evident that multiple yet simple mobile robots can traverse the terrain much faster than using a single monolithic mobile robot [Juliá 2012]. We have further seen in Chapter 4 that most of the terrain coverage algorithms employ some form of dispersion, either local or global, to ensure effective utilization of robots. Local dispersion allows robots to spread out, but confines them within a fixed communication range [Mukhija 2010]. On the other hand, global dispersion spreads out the robots to span much larger areas and aims to gather more information in lesser time. Global dispersion demands either a centralized solution or greater inter-robot communication range. One limitation of the existing approaches which was not addressed until recently is that, these approaches cover the terrain in an irregular fashion [Dasgupta 2009, Puig 2011, Sheng 2006]. They do not consider the usability of the already covered region. For example, in the task of floor cleaning in an office building, these approaches do not ensure the cleanliness and thus usability, of large unbroken regions until the bulk of the coverage task is accomplished. In this chapter, we debate and demonstrate that faster completion of coverage can be attained by using structured trajectory approaches without explicitly targeting dispersion. Following are the major contributions of this chapter:

- We propose, Frontier Allocation Synchronized by Token Passing (FAST), a completely distributed, algorithm for online terrain coverage, that can be deployed in a static terrain and which achieves complete coverage even in the case of (multiple) robot failures, on the assumption that one robot survives till the end.
- FAST covers the terrain in a regular and contiguous fashion, thereby immediately leaving the covered portions of the terrain accessible and usable. Moreover, the recovery protocol in case of robot(s) failure ensures that no information concerning the already covered regions is lost and reorganization time is also less.

- A scheme for dissemination of information is proposed, using which robots can exchange map data by sending and receiving messages. This scheme enables robots to map arbitrarily complex and large terrains and still maintain a low-memory-footprint for communication, with messages having a space complexity of O(N), where N is the number of robots.
- We have proposed, Back Tracking Spiral Algorithm - Coordinated Multi-Robot Improved (BSA-CMI). This is an algorithm for multi-robot coverage and extends a structured trajectory approach suggested in [Gerlein 2011]. It is demonstrated that BSA-CMI works better than many existing approaches. However, it is slower compared to FAST.
- Two distinct approaches [Gautam 2015, Gerlein 2011] representative of the state-of-the-art have been implemented and evaluated, both in simulation and on actual robots. Terrains of varying complexity (in terms of obstacle distribution) have been used. Statistically it is shown that FAST outperforms all the other approaches in terms of coverage completion time and lesser redundant coverage. To the best of our knowledge, such an evaluation across approaches for exploration and coverage has not been performed previously.

In FAST, known frontier cells at a given time instant are chosen to be covered by the robots in a mutually exclusive manner using a token passing mechanism. Each robot maintains a local instance of the global terrain map in the form of an occupancy grid map [Elfes 1989] which is periodically updated. Robots can explicitly send and receive messages to and from their peers. We assume that robots work in a noise-free environment for any communication loss to occur, as the use-case considered here is that of floor cleaning in an indoor environment. However, the robots themselves can fail. A fault-tolerance mechanism is proposed to handle robot failures. The remainder of this chapter is organized into ten sections. The main motivation behind this research is discussed in Section 5.2. A brief discussion on a set of representative approaches for multi-robot terrain coverage that are compared with FAST is presented in Section 5.3. The description of the terrain coverage task and the robot model is presented in Section 5.4 and Section 5.5 respectively. The proposed approach i.e., FAST, is detailed in Section 5.6. Section 5.7 presents the mechanism of handling robot(s) failures. In Section 5.8, we have proposed a weakly centralized algorithm for online terrain coverage referred to as BSA-CMI by extending BSA-CM [Gerlein 2011]. Simulation results and the results of real world experiments across a

spectrum of approaches are discussed in Section 5.9. Chapter summary is presented in Section 5.10.

5.2 RESEARCH MOTIVATION

One limitation of the existing terrain coverage approaches which was noticed in our research is that, many of these approaches cover the terrain in an irregular fashion, without considering the usability of the already covered region. For instance, consider the task of floor cleaning in an office building. These approaches do not guarantee the cleanliness of large unbroken areas until a majority of the task is complete. FAST on the other hand, incrementally traverses the terrain generating structured trajectories for each robot. Following a structured trajectory for coverage path planning is proven to be a very powerful approach in the literature [Gonzalez 2003]. This renders large portions of the terrain usable even before the completion of the coverage task. The map representation techniques used in FAST render it scalable to large terrains, without affecting the volume of communication in the multi-robot system. The distributed nature of FAST allows incorporation of fault-tolerance mechanisms. Empirical investigations on maps of varied complexities and sizes both in simulation and on an experimental test-bed demonstrate that the proposed algorithm performs better than some of the benchmark approaches in terms of coverage completion time and less redundant coverage.

5.3 DISCUSSION ON REPRESENTATIVE APPROACHES

In this section, a set of approaches that are representative of the state of the art in multi-robot terrain coverage and exploration are discussed. We conducted a detailed discussion on [Yamauchi 1998], [Burgard 2005], [Puig 2011], and [Gautam 2015] in Chapter-3. Therefore, in this chapter the discussion is restricted to a single robot structured trajectory approach known as BSA [Gonzalez 2003] and its multi-robot variant BSA-CM [Gerlein 2011].

5.3.1 BSA: A Coverage Algorithm

BSA stands for Backtracking Spiral Algorithm and it works for a single robot [Gonzalez 2003]. It assumes an occupancy grid based decomposition of the environment where size of each cell is equal to the footprint of the robot. Also whole of the free space is assumed to be a single connected component. This algorithm decomposes the free space into more than one structured spiral trajectories. The algorithm ensures complete coverage of all the free cells in the grid map. The main strategy of BSA can be divided into two parts:

- Covering the free grid cells using structured spiral trajectories and creation of simple regions
- Using backtracking mechanism to link various simple regions

The simple regions are incrementally constructed as the robot navigates the free space by following the structured spiral trajectories. All grid cells are initially unknown. When the robot traverses a particular cell it is marked as a "virtual obstacle" which cannot be accessed by the robot as it follows the spiral path. Other cells which are obstructed are marked as "real obstacles". As a result of following a spiral structured trajectory concentric ring like paths are formed that bring forth an unbroken route from the region's periphery to a central point of spiral termination. The robots are initialized close to an obstacle either real or virtual. The obstacle should be located at the Reference Lateral Side (RLS) which refers to the direction in which robots should seek for obstacles while navigating and spiraling in. The opposite of Reference Lateral Side (RLS) is Opposite Lateral Side (OLS). The robots reactively execute Algorithm 5.1, the BSA Coverage algorithm to generate spiral trajectories.

Algorithm 5.1 BSA Coverage

```
1:    if obstacles all around then
2:        backtrack
3:    else if no obstacle in RLS then
4:        turn to RLS
5:        move forward
6:    else if obstacle in front then
7:        turn to OLS
8:    else move forward
9:    end if
```

Some of the important properties of Algorithm 5.1 are listed below:

- When executing the instructions specified by the algorithm virtual obstacles and real obstacles are treated alike.
- When the algorithm terminates the cells that are marked as virtual obstacles represent unobstructed covered cells.
- Along the spiral path while the robot is executing the algorithm backtracking points representing an alternate route are identified and recorded.

- When the algorithm terminates the robot adopts a shortest path to reach the nearest backtracking point. From this point onwards the robot again starts spiraling in.
- The robot stops executing the algorithm when all backtracking points are exhausted from its list. This also indicates that no free and uncovered cells are left.

The execution of the basic BSA algorithm is shown in Figure 5.1. The three spirals are shown with black colored lines. The robot, shown in maroon colored circle, always starts from a cell with an obstacle on its reference lateral side. It can be seen that when the robot is at position# 3, on its reference lateral side are green colored cells that are already covered and are treated as a virtual wall.

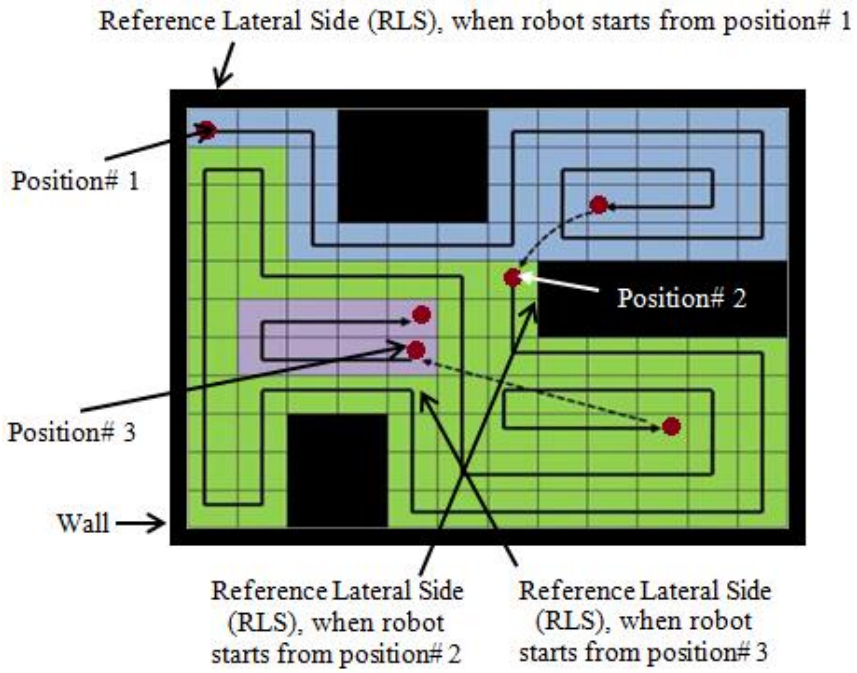


Figure 5.1 Basic BSA Algorithm in Execution

Although the algorithm is very simple and completely covers the terrain, it has a limitation in the sense that the robots are always required to start from a position adjacent to a wall on its reference lateral side. The authors have extended this algorithm for multi-robot a system in [Gerlein 2011] discussed next.

5.3.2 BSA-CM

In [Gerlein 2011], the authors have proposed, Coordinated Multi-Robot Backtracking Spiral Algorithm (BSA-CM) by extending BSA [Gonzalez 2003]. In this algorithm also the robots execute Algorithm 1 suggested for backtracking spiral coverage. In BSA-CM every robot after reaching to the end of its spiral path negotiates with other robots for one of the

remaining backtracking points in a distributed manner using auctions. Following are the crucial properties of this algorithm:

- Every robot knows the initial position of all the other robots.
- Each robot has a local instance of the global map which is updated every time some robot(s) makes a move and a cell(s) is modified. This information is broadcasted to all the robots.
- Robots can be in one of the possible three states i.e., inactive, spiral, or return.
- It is necessary for each robot to know the state of all the other robots. This is also achieved through broadcasts.
- The robot that has won a backtracking point in a most recent auction cannot participate in the subsequent auction. This is done to introduce fairness in the system. But, this requires the robots to keep track of the timings of the auctions.

Conflicts arise as the robots intersect the trajectories of other robots, this particularly happens in corridors and halls when the robots are in return state. This problem is tackled by employing a reservation mechanism wherein each robot under conflict broadcasts its state, current position on the map, next cell it is going to move to and the list of recently discovered backtracking points. There are two limitations of the BSA-CM algorithm. The first limitation is that, each robot has to maintain a lot of information about all the other robots and the environment. Moreover, as the coverage task progresses this information is updated in real-time requiring all robots to continuously send broadcast messages in order to maintain most recent and consistent information about the state of the environment and the state of the other robots. Even in moderate sized maps the volume of communication is going to remain very high. The second limitation is that, the method used by robots for selecting backtracking is purely greedy and does not disperse the robots effectively. This causes a problem in the sense that, when the robots are not dispersed enough, their spirals intersect and they have to backtrack. When backtracking they again end up choosing backtracking points in close proximity of each other eventually requiring the robots to follow redundant trajectories and the process repeats frequently almost after each auction.

5.4 PROBLEM STATEMENT

The task of terrain coverage using a homogeneous team of *n* robots is considered. The terrain is split into a number of equal-size square grid cells, with each such cell having an expanse

equal to the footprint of a robot. Robots are initialized at arbitrary locations on the terrain. Each robot is equipped with sensors that can sense obstacles lying within a space of one cell from the robot. Robots can reliably communicate with each other. By reliable, we refer to the fact that messages sent from a sender definitely reach the receiver, unless the receiver is out of reach. Further, a sender is able to detect whether or not the receiver is reachable. Formally, the terrain coverage task is stated as follows:

Given N robots and a bounded terrain T composed of grid cells, assign a finite set of moves to each robot in a manner that each grid cell in T is visited by at least one robot, and within a finite number of time steps.

5.5 ROBOT MODEL

Each robot in the multi-robot team is modeled as a finite-state machine (FSM) as shown in Figure 5.2. The team can then be viewed as a set of equivalent and independent FSMs that interact by passing messages and are made to function in a synchronized fashion by using a special message referred to as the token. The FSM model of each robot has five states, viz. *Sense Neighboring Cells, Wait for Token, Critical Section, Move Towards Target,* and *Handle Timeout*. In the *Sense Neighboring Cells* state, the robot fires its sensors and determines the occupancy of its neighboring cells (if the occupancy is not already known to the robot) and transits to the *Wait for Token* state. In this state, the robot waits until it receives the token. If the token is not received until a specified deadline, a timeout is said to have occurred. The robot handles this timeout by transiting to the *Handle Timeout* state, after which it returns to the *Sense Neighboring Cells* state. In case the timeout does not occur, the robot transits from the *Wait for Token* state to the *Critical Section* state after receiving the token. At any particular time instant only one robot from the team is allowed to be in the *Critical Section*. Frontier selection and map updates are performed by the robots in the *Critical Section*. After releasing the token, the robot transits to the *Move Towards Target* state, where it takes one step closer to the target cell it has chosen, after which the robot returns to the initial state, viz. *Sense Neighboring Cells*.

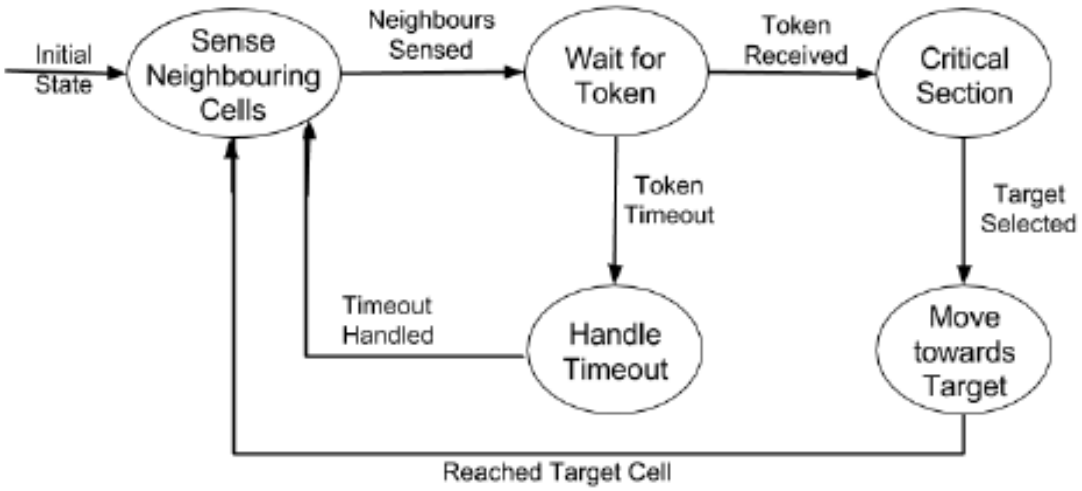


Figure 5.2 Finite State Machine model of a robot

5.6 FAST: SYNCHRONIZED FRONTIER ALLOCATION

In this section, we detail, FAST (Frontier Allocation Synchronized by Token passing) to synchronize the allocation of a unique frontier cell to each robot in a particular time step.

5.6.1 The Need for Synchronization

The problem of terrain coverage is equivalent to that of allocating frontier cells, until all frontiers are exhausted. Based on the taxonomy presented in [Gerkey 2004a], the frontier cell allocation problem considered here falls under the ST-SR-IA (Single Task - Single Robot - Instantaneous Assignment) category of multi-robot task allocation problems. Here, each robot can take up the task of visiting only one frontier cell at a time (Single Task) and each frontier cell can only be allotted to one robot (Single Robot). This (re)allocation of frontier cells to robots occurs at each time step (Instantaneous Assignment). Moreover, robots perform target selection locally. To ensure that multiple robots do not select the same frontier, synchronized frontier selection is necessary.

5.6.2 Synchronized Frontier Selection

In FAST, the multi-robot system is analogous to a distributed computing system, where synchronization and mutual exclusion are achieved through token passing. Each token has a sequence number that serves as a timestamp to distinguish new tokens from old ones. A token also carries the map updates with respect to a global map which is shared by all the robots and is used for synchronized frontier selection. Each robot is assigned a unique integer id, referred to as the rank. The robots set up a virtual token ring network, with the

robot having the lowest rank assuming the role of a monitor. A token is circulated in the network, starting from the monitor, and reaches all robots in the increasing order of their ranks. The only exception to this rule is the monitor, which receives the token from the robot with the highest rank in the ring. The underlying physical topology of the (wireless) network allows for the transmission of broadcast messages. The information carried by the token is shown in Figure 5.3. Each token contains a sequence number, which indicates the recency of the token, and the ranks of the sending and the receiving robots. In addition to this, the token carries a compact representation of the map updates that is discussed in Section 5.5.4.

Figure 5.3 Structure of the Token

The algorithm that each robot runs is outlined in Algorithm 5.2. Each robot could be executing one of the three sections of this algorithm, viz. entry section, critical section, or remainder section. In its entry section, each robot waits for a token until a predetermined timeout period (Algo. 5.2, lines 3-4). Only after receiving the token, it can enter its critical section. If the token is not received before the timeout, a failure is identified and is handled by calling the *IssueBeacon* procedure (Algo. 5.2, lines 5-9). The exact mechanism of handling failures is described in Section 5.7. The selection of a frontier cell is to be made by each robot in its critical section, using the *SelectNextTarget* procedure (Algo. 5.2, line 16). Each robot maintains a local instance of the global map, which is updated and exchanged with other robots, along with the token. Hence, map updates are also done in the critical section (Algo. 5.2, lines 14, 17). The robot then exits the critical section, and in the remainder section, makes a move that takes it one cell closer to its selected frontier cell (Algo. 5.2, lines 18-19). The algorithm terminates when no more frontier cells are visible to the robot (Algo. 5.2, line 20).

Algorithm 5.2 FAST

```
1:      {Algorithm to be run on each robot}
2:      repeat
3:          start token timer with timeout τ1                ► Entry section
4:          while token not received do
5:              if tokenTimeout then
6:                  stop token timer
7:                  call IssueBeacon()
```

```
8:                          goto line 3
9:                  end if
10:                 if beaconReceived then
11:                         call AcknowledgeBeacon()
12:                 end if
13:         end while
14:         read token and update map                          ► Critical section
15:         update token sequence number and frontier cell list
16:         call SelectNextTarget()
17:         flag selected target on token as allotted
18:         release token
19:         move a step closer to target                       ► Remainder section
20:     until no more frontiers are reported on the map
```

The monitor is responsible for the creation and maintenance of the token. Initially, the monitor generates a token, updates the sequence number of the token, and uses the token to select a frontier cell. It then releases the token and performs its move. The token, then circulates in the network.

5.6.3 Synchronized Move Selection

Let the coordinates of a given robot in the global frame of reference be r. Also, let θ_P be the global preferred direction. The direction in this context does not have any real world reference or sensor measurement. It only refers to the direction of the map. For Example, North in a grid based map would be the direction in which the row coordinates decrease and column coordinate remains unchanged. The preferred direction is corresponding to the global from which usable area is to be created, and is common for all the robots. For example, if the preferred direction is set to cover north, all robots begin coverage by proceeding to the north edge. After reaching the north edge, they keep moving down, covering each row, until they reach the south edge. The distance ρ from the robot to any cell with coordinates c, is defined in equation 5.1 as

$$\rho = path(r, c) \tag{5.1}$$

where $path(r, c)$ denotes the length of the shortest path between robot r and cell c. While computing the shortest path, the locations of known obstacle cells are also taken into account by the path function. Also, for a cell c, we define the direction of c, denoted by θ_c, as the orientation of the position vector of the cell with respect to the global X axis.

In its critical section, as shown in Algorithm 5.2, a robot examines the map it received with the token and updates its local map instance (Algo. 5.2, line 14). It then increments the sequence number of the token and updates its list of newly discovered frontier cells (Algo. 5.2, line 15). A new frontier cell is selected by calling the *SelectNextTarget* procedure as shown in (Algo. 5.2, line 16). The selected cell is flagged with the robot's rank (Algo. 5.2, line 17). In the *SelectNextTarget* procedure shown in Algorithm 5.3, a robot examines the frontier cells in its updated local map and makes a list L of the closest un-allotted frontier cells (Algo. 5.3, line 2). It then examines all frontier cells $F \in L$. If any cell F is found with $\theta_F = \theta_P$, that cell is selected (Algo. 5.3, line 3). If no cell satisfies this criteria, the robot computes $\delta = |\theta_F - \theta_P|$ for each $F \in L$ (Algo. 5.3, lines 4-5). The frontier cell with the least value of δ is selected (Algo. 5.3, line 6). Ties are broken arbitrarily. If a robot is not allotted any frontier cell, but it sees that the map contains some frontier cells allotted to other robots, it selects its current location as its target position.

Algorithm 5.3 SelectNextTarget

1: make a list L of closest un-allotted frontiers
2: **if** $\exists F \in L$ such that $\theta_F = \theta_P$, select F
3: **if** $\nexists$ such F **then**
4: $\forall F \in L$, compute $\delta = |\theta_F - \theta_P|$
5: select the first frontier cell with the least δ
6: **end if**

5.6.4 Map Representation

In FAST, the instance of the map of the environment stored by each robot is represented by a thresholded occupancy grid, with the following discrete states - *visited, flagged, unvisited, frontier, occupied*. In most of the existing distributed algorithms for terrain coverage, there is also the notion of a *global map* [Sheng 2006]. The global map integrates the data from all local maps into a single consistent representation. However, in FAST, there is no single global map. There exist only map updates that are generated in the following fashion. Let N be the number of robots and M_i^{t-1} be the local map of robot i at the end of iteration $t-1$. In iteration t, robot i scans its neighboring cells and updates its local map. Upon receiving the token, the robot performs further map updates, followed by frontier selection (the details of which are presented in the following section). Then, the robot updates the map by marking

its selected frontier cell. Let this map be denoted by M_i^t. Now, the robot performs an exclusive-OR of M_i^{t-1} with M_i^t as shown below:

$$U_i^{(i+1)\%N} = \begin{cases} M_i^{t-1} \oplus M_i^t & ,if \quad t > 1 \\ M_i^1 & ,if \quad t = 1 \end{cases} \tag{5.2}$$

The *XOR* operation shown in equation 5.2 will leave only those frontier cells whose state has changed in the previous iteration. The set of these cells and their corresponding states $U_i^{(i+1)\%N}$, where % denotes the modulus operator, is the set of map updates passed from robot *i* to its *Nearest Downstream Neighbor* (*NDN*). The robot then passes these map updates along with the token. Since all robots follow this process, and because the token is drawn in a synchronized fashion, the local map of each robot is consistent with the global position, albeit with a time lag of one iteration.

This method of passing map updates also results in the invariance of the space complexity of the message with the size of the map. In any particular iteration (except the first iteration), each robot can find at most seven frontier cells. This is reasoned as follows. A cell C_{ij} can have at most eight neighboring frontier cells. Taking for granted that it is not the first time instant, the robot must have moved into C_{ij} from some other cell. But, to reach C_{ij}, the robot must pass through at least one of the neighbors of C_{ij}, which leaves at most seven neighboring cells as frontier cells. Further, a robot can mark one cell as flagged and one cell as visited. Hence, in the worst case, the update set $U_i^{(i+1)\%N}$ can contain a maximum of *9N* values. Since this update set occupies the most of the space of the token, the space complexity of messages passed is linear in the number of robots. Since this complexity is independent of the size of the map, FAST is scalable to large terrains.

5.6.5 Map Updates

When a robot receives a token and enters its critical section, it scans the token and updates the map (Algo. 5.2, line 14). Precisely speaking, the following marking operations are performed by robot *i* on its local map and on the map updates carried along with the token.

- The frontier cells that are already selected by other robots who have received the token before robot *i* are marked as flagged.
- The sequence number of the token is incremented.

- The frontier cells that robot *i* moved to just before it began the current iteration is marked as visited.
- The new frontier cells discovered by robot *i* in the course of the movement are marked as frontiers.
- The cells that all robots *j (j≠i)* which have moved to in the previous iteration are marked as visited.
- The new frontier cells discovered by all robots *j (j≠i)* are marked as frontiers.

The robot then chooses its next frontier cell by calling the *SelectNextTarget* procedure as shown in Algorithm 5.2 and 5.3. It again updates the map by marking as flagged the frontier cell chosen by the *SelectNextTarget* procedure.

5.7 FAULT TOLERANCE AND ROBUSTNESS

In this section, we present the fault-tolerant mechanisms incorporated in FAST that makes it robust to failure of one or more robots.

5.7.1 Network Setup

As mentioned in the previous section, the logical robot-to-robot network topology is modeled as a variant of the token ring topology. In this topology, robots pass messages in a circular fashion, with each robot sending messages to the next robot in the ring, referred to as its *Nearest Downstream Neighbor (NDN)*, and receiving messages from the previous robot in the ring, referred to as its *Nearest Upstream Neighbor (NUN)*. When a monitor sets up a network, it also passes the network configuration information to every robot in the network. This configuration information specifies the rank of each robot, and helps determine the *NUN* and the *NDN* for each robot. It must be noted that this is only a logical topology used for inter-robot communication and should not be confused with the underlying physical topology.

5.7.2 Fault-Tolerance Mechanism

There are two kinds of failures in the token ring topology considered in this work - token loss, and robot failure. Since reliable inter-robot communication is assumed in our setup, token loss does not occur unless some robot fails while holding the token. In our fault-tolerance mechanism, we assume that one or more robots can simultaneously fail. Moreover, a failed robot may or may not be the monitor of the token ring. In FAST, both single and

multi-robot failures are handled using the same mechanism, as shown in Algorithm 5.4 and 5.5. Each robot has complete knowledge of the network topology (by virtue of the configuration information received from the monitor, during the network setup phase). In their entry sections, all robots start of a timer, referred to as the token timer, which counts down to zero from a predetermined initial value τ_1 (Algo. 5.2, line 3). This value τ_1 is set to the maximum time (determined empirically in Section 5.9.2) in which any robot should definitely receive a token back, after releasing it. In the event of a failure, some robot will experience a timeout of the token timer. This robot will then broadcast a special signal, referred to as the beacon signal (Algo. 5.4, line 1) and wait for a stipulated amount of time τ_2 (determined empirically in Section 5.9.2) for all the responses (Algo. 5.4, line 2).

Algorithm 5.4 IssueBeacon

```
1:    transmit a beacon signal
2:    gather all received acknowledgements upto time τ2
3:    if monitorIsAlive then
4:        send results to monitor
5:        start beacon timer with timeout period τ2
6:        while new network not set up do
7:            if beaconTimeout then
8:                goto line 2
9:            end if
10:       end while
11:       {network setup successful}
12:   else
13:       become monitor
14:       set up network of alive robots
15:   end if
16:   receive map from NAUN
17:   {resume normal operation}
18:   return
```

When other robots receive this beacon signal, they call the *AcknowledgeBeacon* procedure (Algo. 5.2, lines 10-12). Their token timers stop, and another timer, known as the beacon timer, starts on each robot (Algo. 5.5, lines 1-2). These robots then send an acknowledgement to the beacon sender (Algo. 5.5, line 3). This acknowledgement carries the rank of the robot and the sequence number of the most recently received token. After collecting all the responses, the beacon sender determines if the monitor is alive by

examining the responses (Algo. 5.4, line 3). If the monitor is alive, the beacon sender sends all the responses to the monitor and starts a beacon timer (Algo. 5.4, lines 4-5). The monitor then establishes the token ring using the ranks of each respondent. Now, the beacon sender receives the token from the *Nearest Active Upstream Neighbor (NAUN)* identified by the monitor (Algo. 5.4, lines 16-17). All robots clear their beacon timers and resume normal operation.

Algorithm 5.5 AcknowledgeBeacon

```
1:    stop token timer
2:    start beacon timer with timeout τ2
3:    send acknowledgement to beacon sender
4:    while new network not set up do
5:        if beaconTimeout then
6:            wait for (n * τ2) / 2    (milliseconds), where n = rank of the robot
7:            call IssueBeacon()
8:        end if
9:        if beaconReceived then
10:           goto line 3
11:       end if
12:   end while
13:   {resume normal operation}
14:   return
```

If the monitor is not found to be alive, the node that first experienced a token timeout and hence transmitted the beacon signal assumes the role of the monitor and sets up the network (Algo. 5.4, lines 13-14). The beacon timer is initialized to a value that would suffice for all acknowledgement transmissions, and new token ring setup time. The token timer is initialized to a value that would suffice for all token transmissions and robot movements. Whenever a beacon timer expires, a robot waits for some finite time proportional to its rank and issues a beacon signal (Algo. 5.5, lines 5-8). This is done to ensure that at any time instant; only one robot issues a beacon signal. An important aspect to ensure is the correctness of a token after the network is re-established. It is needed so that no frontier cell is left unvisited. We have seen that when a robot fails, the token that is circulated in the new network is the version of the token with the highest sequence number. To ensure complete coverage, we need the frontier cells that are left unvisited, since they were chosen by a robot which has failed to visit them. However, since the robot has failed, we have no way of

knowing whether or not the chosen frontier was actually visited. In FAST, we assume that it has not been visited. To enforce this assumption, the monitor that re-establishes the network clears the marked cells in the map that were chosen by the robots which have failed. Although this strategy may lead to redundancy, it ensures complete coverage even in the face of failure of robots.

5.8 BACKTRACKING SPIRAL COVERAGE – COORDINATED MULTI-ROBOT IMPROVED (BSA-CMI) ALGORITHM

In this section we propose, Coordinated Multi-Robot Backtracking Spiral Algorithm Improved (BSA-CMI) which is an improvement over BSA-CM proposed in [Gerlein 2011]. In particular, we have addressed the second limitation of BSA-CM which is that of redundant coverage due to local dispersion (details are in Section 5.3.2). In order to ensure faster completion of coverage, in BSA-CMI, the selection criteria of backtracking point is modified. This approach is weakly centralized in the sense that, a central station maintains a global list of backtracking points and provides access of this list to the robots, whilst ensuring mutual exclusion. Also, each robot receives map updates at the end of every time step from every other robot. Whenever any robot reaches the center of its spiral, it contacts the central station to access the list of backtracking points. The proposed algorithm associates with each backtracking point a constant penalization value. The cost of selecting a backtracking point is then defined as the sum of the distance to the backtracking point from the robot's current position and the penalization value. The backtracking point with the least cost (nearest in terms of Manhattan distance) is chosen. The chosen point (say Δ) is removed from the global list of backtracking points so that no other robot selects the same point. Further, all backtracking points within the radius $\sigma = 5$ (an empirically determined constant) of Δ are penalized by a high constant value $\delta = 100$. This causes the robots to disperse and ensures that not many robots are close to each other. Hence their spirals do not intersect. A high value of σ does not allow some robots to select a backtracking point rather they will be pulled by random cells in the free space that are still uncovered. Therefore, these robots will have to travel longer distances and still they will contribute to increased redundant coverage. When compared with BSA-CM we have got improved results for BSA-CMI in terms of reduced time to complete coverage and reduced overlapping coverage. The spiraling algorithm used in BSA-CMI is exactly the same as the one used in BSA-CM. The simulation and experimental results are presented in Section 5.9.

5.9 EXPERIMENTS AND RESULTS

The proposed approach has been evaluated extensively in simulation as well as on real multi-robot test-bed. In this section, we provide a description of the experiments performed and analyze the results obtained.

5.9.1 Simulation Results

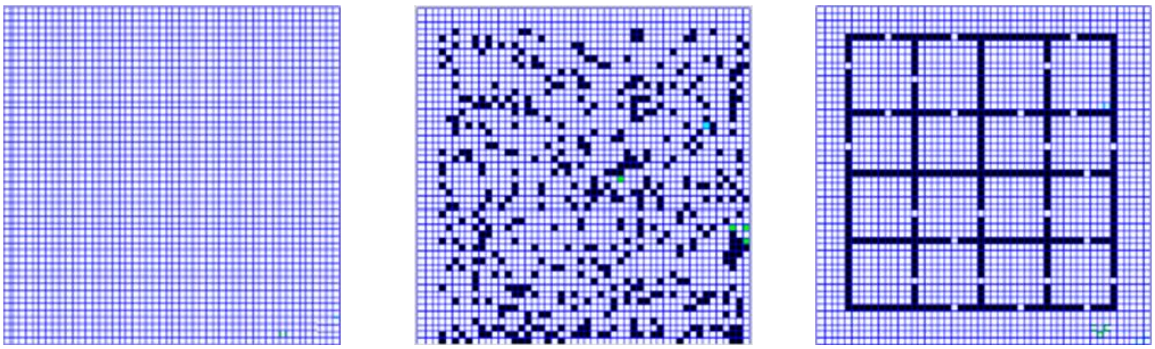
To evaluate the performance of FAST, we consider two benchmark approaches for multi-robot terrain coverage [Gautam 2015, Gerlein 2011]. In Chapter 4, we have proposed a centralized approach, referred to as CAC, for online multi-robot terrain coverage. CAC outperforms three state of the art approaches [Yamauchi 1998], [Burgard 2005], and [Puig 2011]. Therefore, CAC is considered as a representative of approaches that perform frontier allocation followed by re-planning. A structured trajectory based online multi-robot coverage approach is suggested in [Gerlein 2011], referred to as BSA-CM. In this chapter a modified approach referred to as BSA-CMI is proposed by extending BSA-CM. Both BSA-CM and BSA-CMI are considered as a representative of structured trajectory approaches. We have compared FAST with CAC, BSA-CM, and BSA-CMI to establish it as a new benchmark.

All the approaches have been evaluated on maps of different sizes, complexities, and varied team size of the multi-robot system. To provide a consistent comparison among all the approaches, a multi-robot simulation framework is developed in python. In all our experiments the robot's sensing range is restricted to one cell i.e., the eight cells surrounding the robot's current position. The empirical performance evaluation is based on the following metrics:

- *Completion time* - It is measured in terms of total *number of iterations* for complete coverage of the terrain. An *iteration* is defined as the time period in which each robot of the team chooses a cell and moves to it.
- *Percentage redundancy* - It is measured in terms of occupancy grid cells. If there are X number of free cells to be covered and the sum of the total number of moves made by each robot is Y, such that, $Y > X$, then extra moves made by the robot team is $\Delta X = Y-X$. Then percentage redundancy is computed as follows:

$$\% \text{ Redundancy} = [(X + \Delta X) / X] * 100 \quad (5.3)$$

The maps used for evaluating the algorithms are shown in Figure 5.4. Figure 5.4(a) is a Free World map without any obstacles. Figure 5.4(b) is an Outdoor map with many small obstacles arbitrarily scattered in the whole region. Figure 5.4(c) represents an Office type environment wherein there are many rooms with door openings connecting them to other rooms and corridors. The obstacles in Office map are long walls spanning across whole of the environment. The three maps shown in Figure 5.4(a), 5.4(b), and 5.4(c) are large maps with a dimension of 50×50 cells. Also, one important property of the three maps shown in Figure 5.4 is that, the entire free space is a single connected component. Figure 5.5, 5.6, and 5.7 show snapshots of the progress of FAST on different terrains in a simulation environment. In all simulations, the preferred direction was set to North (the top edge of the map). The emergent behavior of creating large, usable chunks of connected visited cells can be observed.



(a) Free World (b) Outdoor Map (c) Office Map

Figure 5.4 Three different terrains used for simulation (dimensions: 50 x 50 cells)

For each map and different sizes of the robot team, the simulation was run 25 times, with random initial positions of the robots. In each case, the time in terms of number of iterations for complete coverage and percentage redundancy is measured. As a general trend it can be observed for all the approaches considered for evaluation that, irrespective of the map, if the number of robots are increased the completion time starts reducing. This is shown in Figures 5.8, 5.10, and 5.12. On the other hand, the percentage redundancy starts increasing as shown in Figures 5.9, 5.11, and 5.13. Next, we discuss the consolidated results of the completion time graphs shown in Figures 5.8, 5.10, and 5.12 and percentage redundancy shown in Figures 5.9, 5.11, and 5.13.

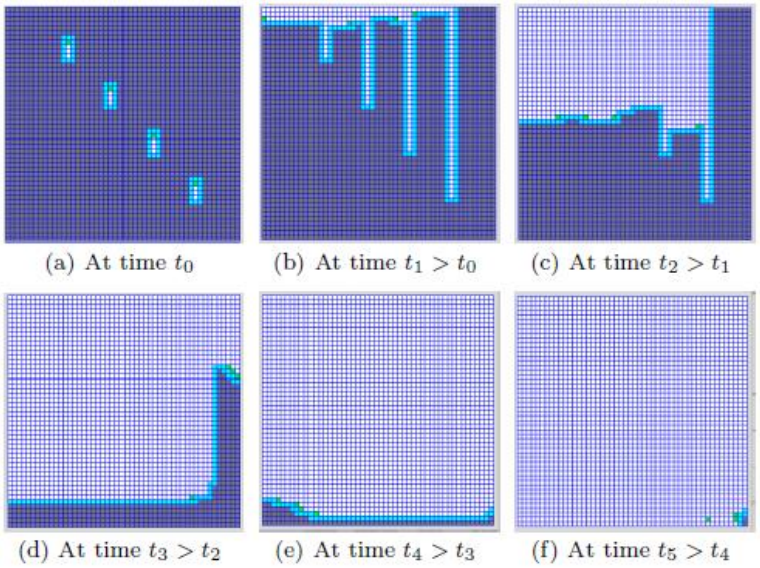


Figure 5.5 Progress of the algorithm in the Free World map with 4 robots

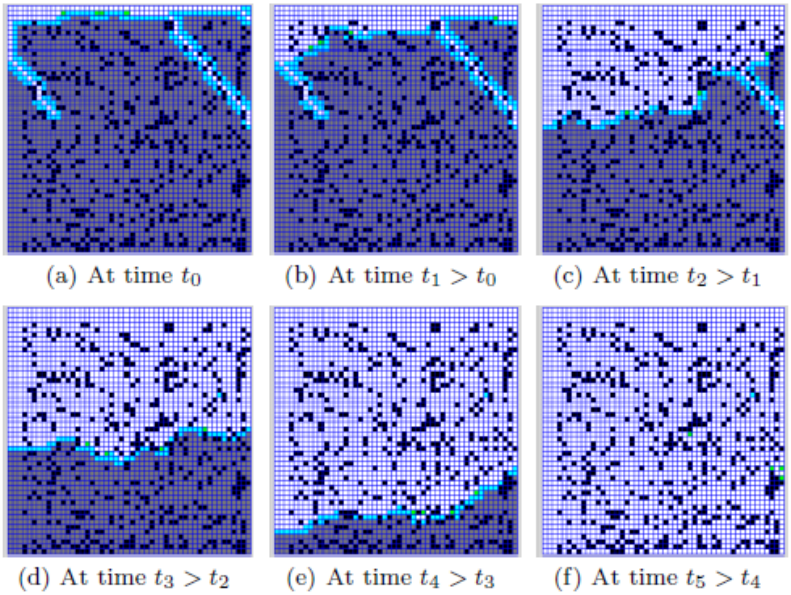


Figure 5.6 Progress of the algorithm on the Outdoor map with 4 robots

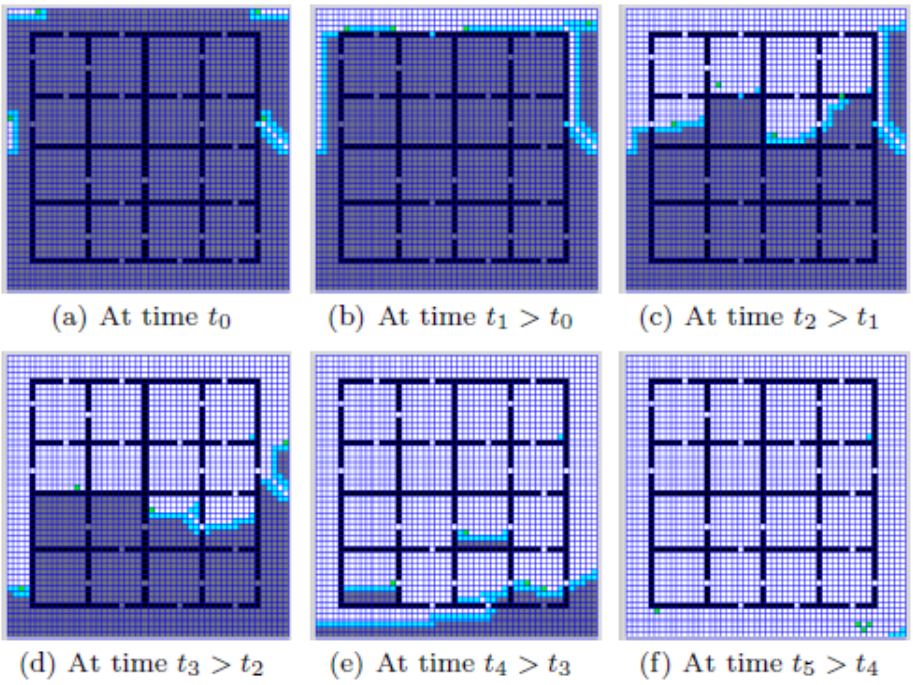


Figure 5.7 Progress of the algorithm in the Office map with 4 robots

From Figures 5.8-5.13 it can be inferred that CAC does not perform better than any of the structured trajectory approaches viz., BSA-CM, BSA-CMI, and FAST. BSA-CM on the other hand, generates structured trajectories and does not leave behind uncovered cells and hence it performs better than CAC. Since BSA-CMI ensures better selection of next target cell by robots when compared to BSA-CM, it performs better, especially if the number of robots are more. FAST performs better than all the other algorithms since it exploits the best of the both worlds, a motion planning scheme which ensures that at each time instant (except when the number of frontier cells are less than the number of robots), every robot has a frontier to move to, and also it follows a structured trajectory. Some other important observations are as follows:

- Both the completion time and the percentage redundancy for all the approaches increase as the complexity of the map increases. This is evident from the completion time graphs of Figure 5.8, 5.10 and 5.12 and redundancy graphs of Figure 5.9, 5.11, and 5.13.
- The percentage improvement (in terms of coverage completion time and non-overlapping coverage) of FAST over other approaches is shown in Tables 5.1, 5.2, and 5.3 with respect to Free World, Outdoor and Office Maps respectively. It can be easily inferred that FAST outperforms all the approaches irrespective of the complexity of the map used for evaluation.

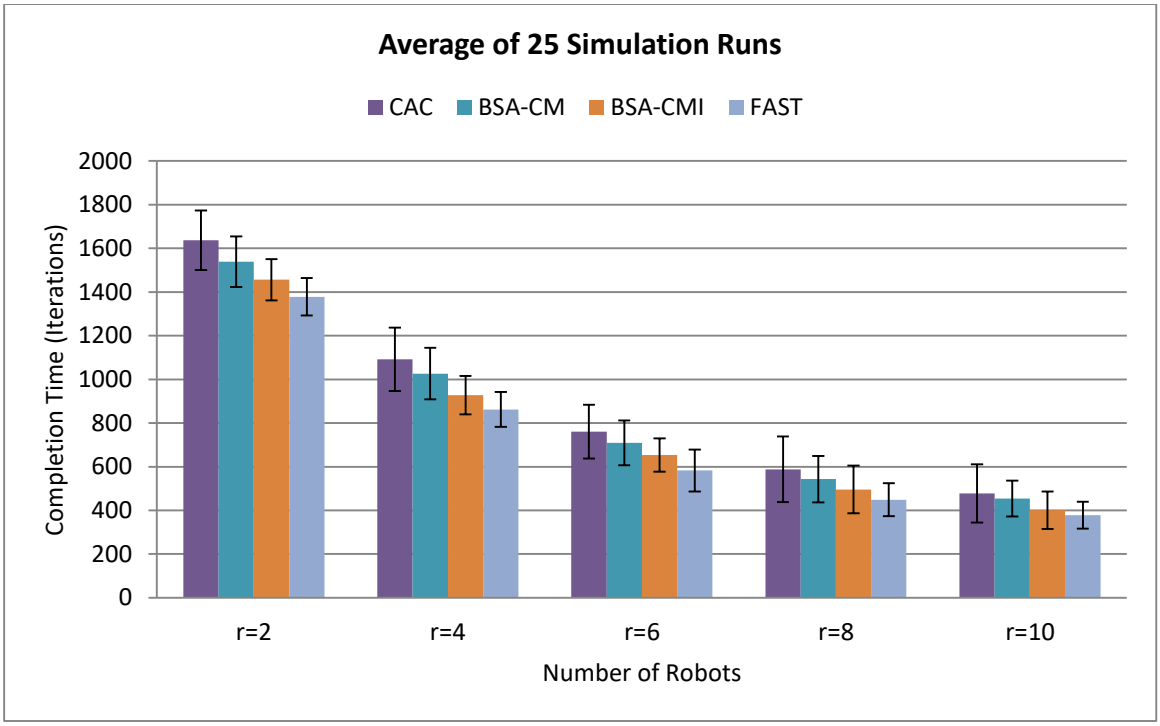


Figure 5.8 Completion Time measured in the Free World Map

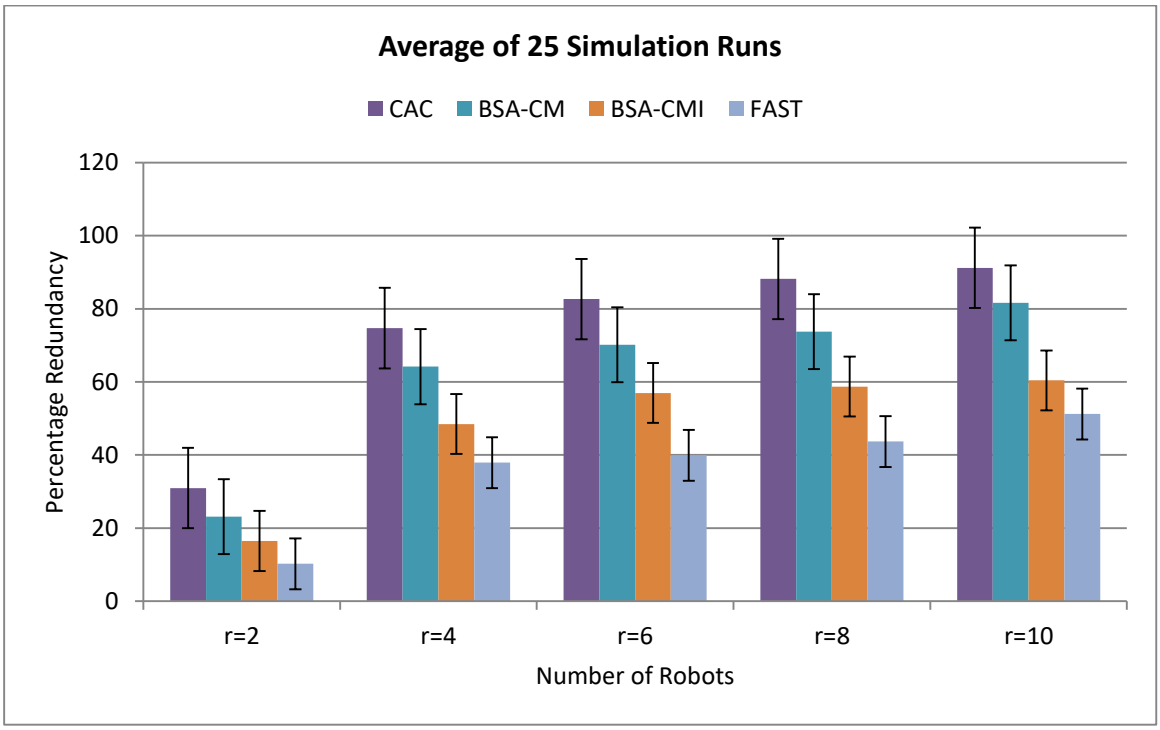


Figure 5.9 Percentage Redundancy measured in the Free World Map

Table 5.1 Comparison of FAST with different approaches in Free World Map

MAP	ROBOTS	FAST vs.	PERCENTAGE IMPROVEMENT IN COVERAGE COMPLETION TIME	PERCENTAGE IMPROVEMENT IN NON-OVERLAPPING COVERAGE
FREE WORLD MAP	2	CAC	15.8	66.9
		BSA-CM	10.4	55.7
		BSA-CMI	5.3	37.8
	4	CAC	21.1	51.7
		BSA-CM	15.9	43.1
		BSA-CMI	7.1	29.9
	6	CAC	23.4	50.5
		BSA-CM	17.3	40.8
		BSA-CMI	10.8	25.6
	8	CAC	23.6	49.3
		BSA-CM	17.7	38.7
		BSA-CMI	9.4	21.7
	10	CAC	20.9	43.9
		BSA-CM	16.7	35.2
		BSA-CMI	5.7	15.2

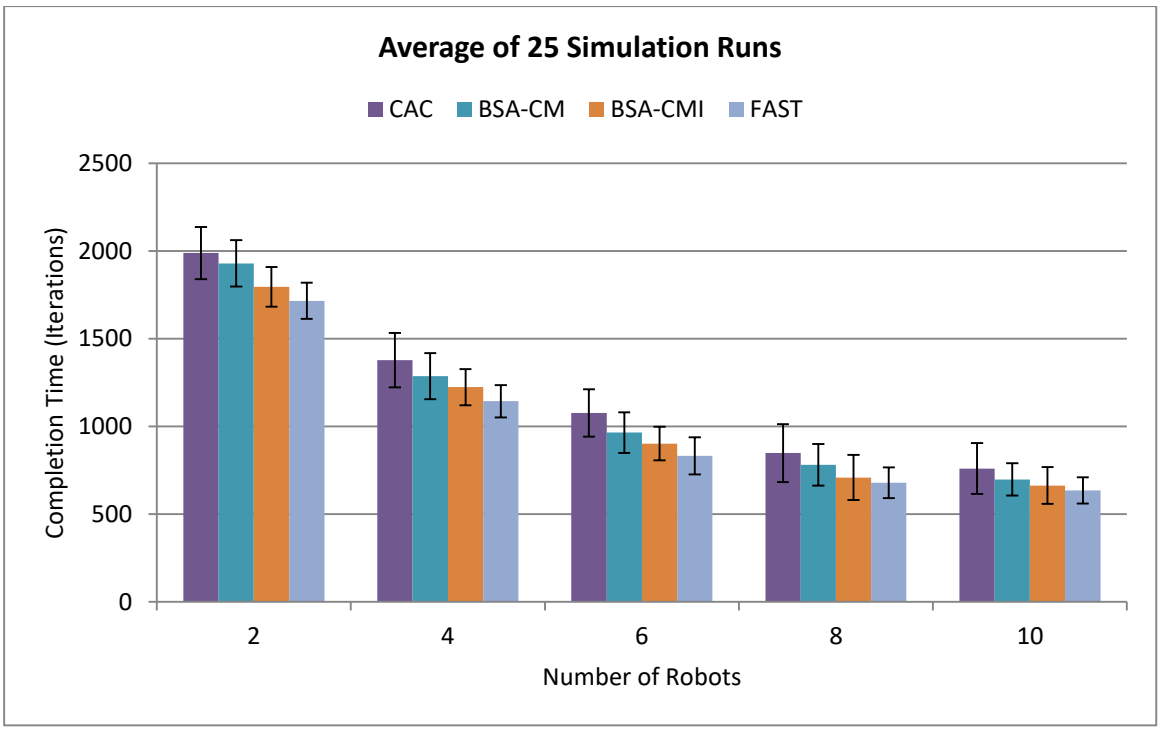


Figure 5.10 Completion Time measured in the Outdoor Map

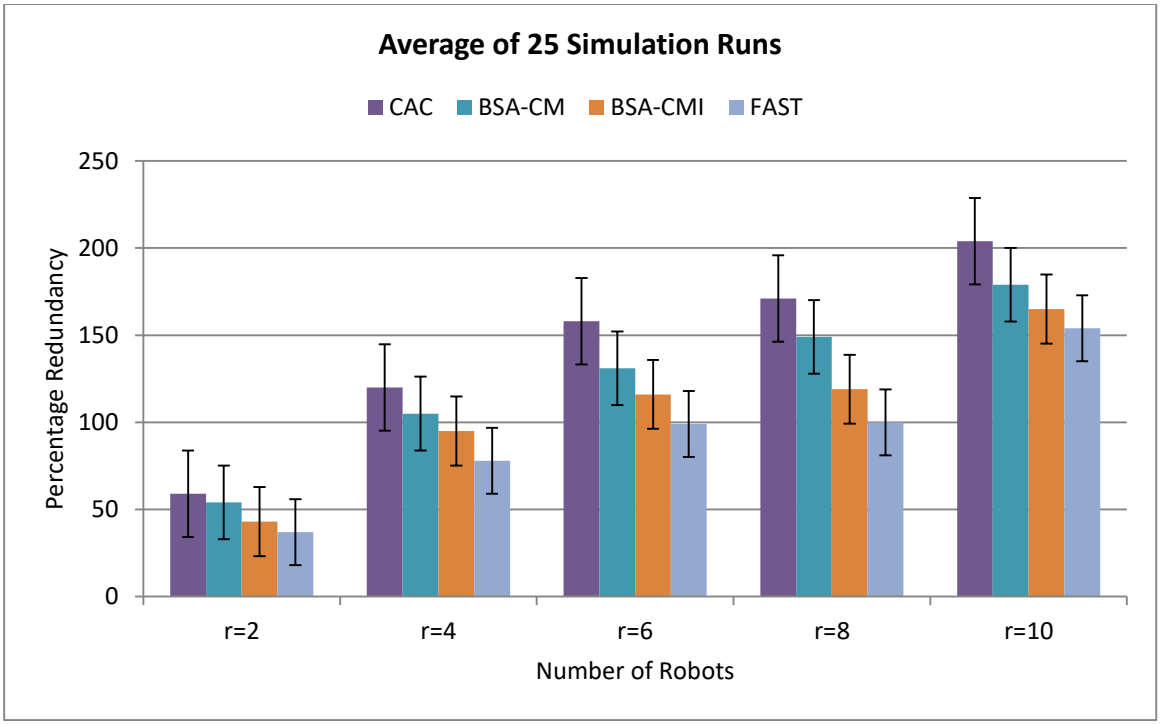


Figure 5.11 Percentage Redundancy measured in the Outdoor Map

Table 5.2 Comparison of FAST with different approaches in Outdoor Map

MAP	ROBOTS	FAST vs.	PERCENTAGE IMPROVEMENT IN COVERAGE COMPLETION TIME	PERCENTAGE IMPROVEMENT IN NON-OVERLAPPING COVERAGE
OUTDOOR MAP	2	CAC	13.7	37.3
		BSA-CM	11.04	31.48
		BSA-CMI	4.4	20.63
	4	CAC	17	35.8
		BSA-CM	12.63	32.88
		BSA-CMI	6.53	14.65
	6	CAC	22.7	33.3
		BSA-CM	13.78	24.42
		BSA-CMI	7.76	13.95
	8	CAC	19.9	31.12
		BSA-CM	13.06	20.95
		BSA-CMI	6.23	12.63
	10	CAC	16.4	24.5
		BSA-CM	9.02	13.96
		BSA-CMI	4.22	6.66

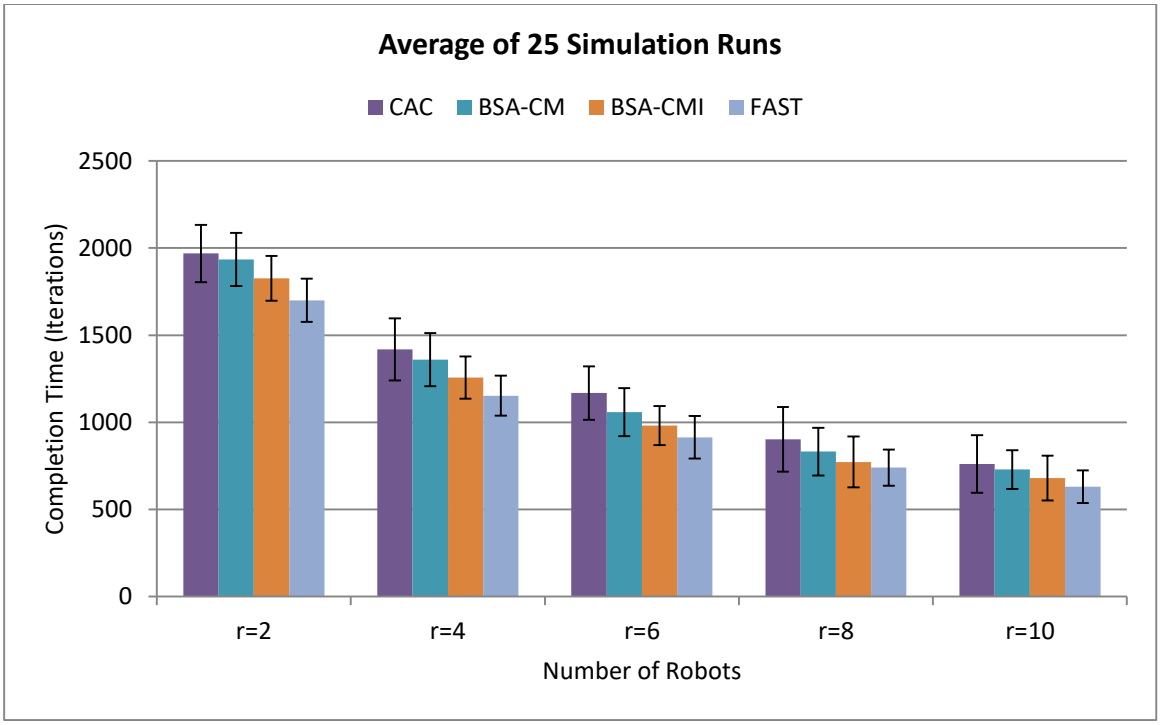


Figure 5.12 Completion Time measured in the Office Map

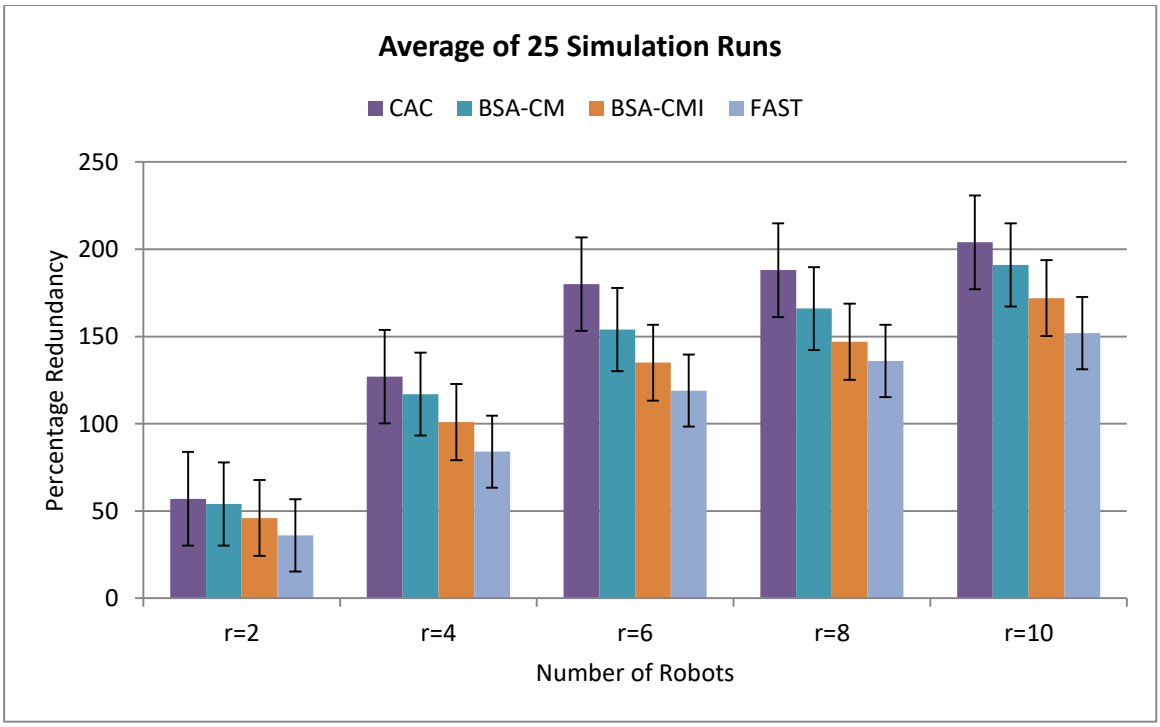


Figure 5.13 Percentage Redundancy measured in the Office Map

Table 5.3 Comparison of FAST with different approaches in Office Map

MAP	ROBOTS	FAST vs.	PERCENTAGE IMPROVEMENT IN COVERAGE COMPLETION TIME	PERCENTAGE IMPROVEMENT IN NON-OVERLAPPING COVERAGE
OFFICCE MAP	2	CAC	13.85	36.8
		BSA-CM	12.47	33.33
		BSA-CMI	6.82	21.73
	4	CAC	18.7	33.9
		BSA-CM	13.69	28.2
		BSA-CMI	6.9	16.83
	6	CAC	21.7	31.7
		BSA-CM	15.22	22.72
		BSA-CMI	8.27	11.85
	8	CAC	18.74	27.7
		BSA-CM	13.44	18.07
		BSA-CMI	7.2	11.62
	10	CAC	17.1	25.5
		BSA-CM	11.65	20.41
		BSA-CMI	4.14	9.48

We now consider another advantage of FAST, i.e., its ability to handle the failure of one or more robots. To validate the same, simulations were performed (on the Free World, Outdoor and Office maps) where one robot from the team was randomly selected and stopped (indicating failure) in its current cell after every 250 iterations. FAST, [Burgard 2005], and [Yamauchi 1998] have been evaluated in this respect. Other approaches i.e., CAC, BSA-CM, and BSA-CMI were not evaluated, as they do not make any claims of fault-tolerance. It can be seen from Figure 5.14, 5.15 and 5.16 that FAST completes coverage in lesser number of iterations as compared to the other approaches, even when the robots fail. It is because the algorithm FAST follows structured trajectories and do not leave unvisited frontier cells in a haphazard fashion, allowing other robots to take over the remaining work of the failed robot(s). In a nutshell, FAST achieves faster coverage compared to all approaches considered for evaluation and for all the maps, robot team sizes, and random distribution of robots. Also, FAST performs less redundant coverage than the other approaches because, in each iteration, every robot makes a move, and in most cases, the move results in robots moving to uncovered cells.

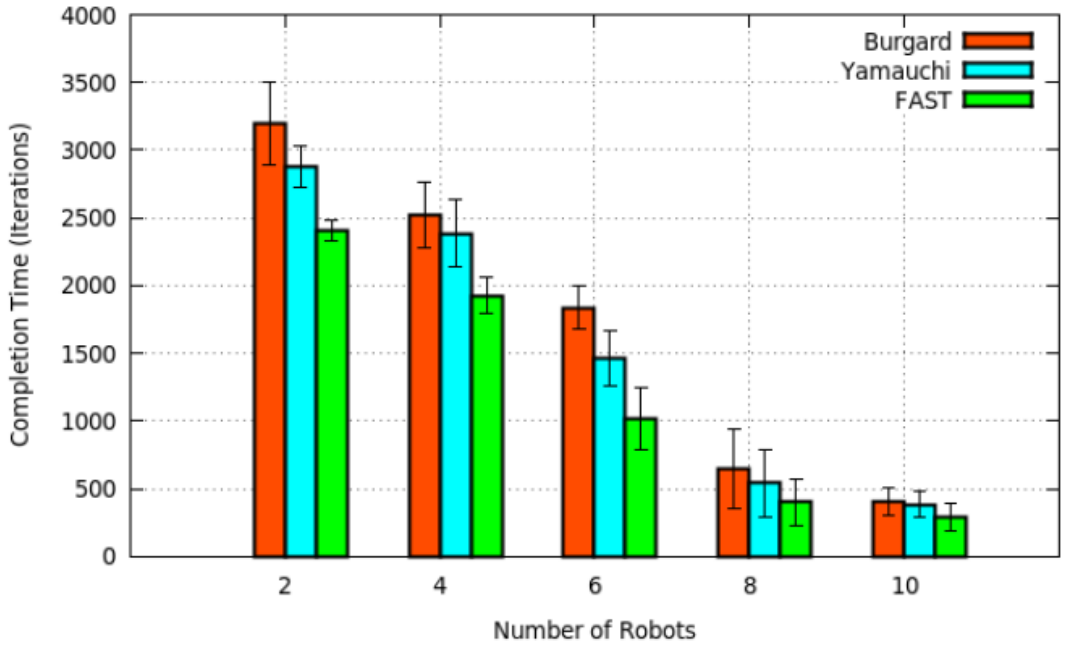


Figure 5.14 Completion time measured in the Free World map with one robot failing every 250 iterations (until only one robot is alive)

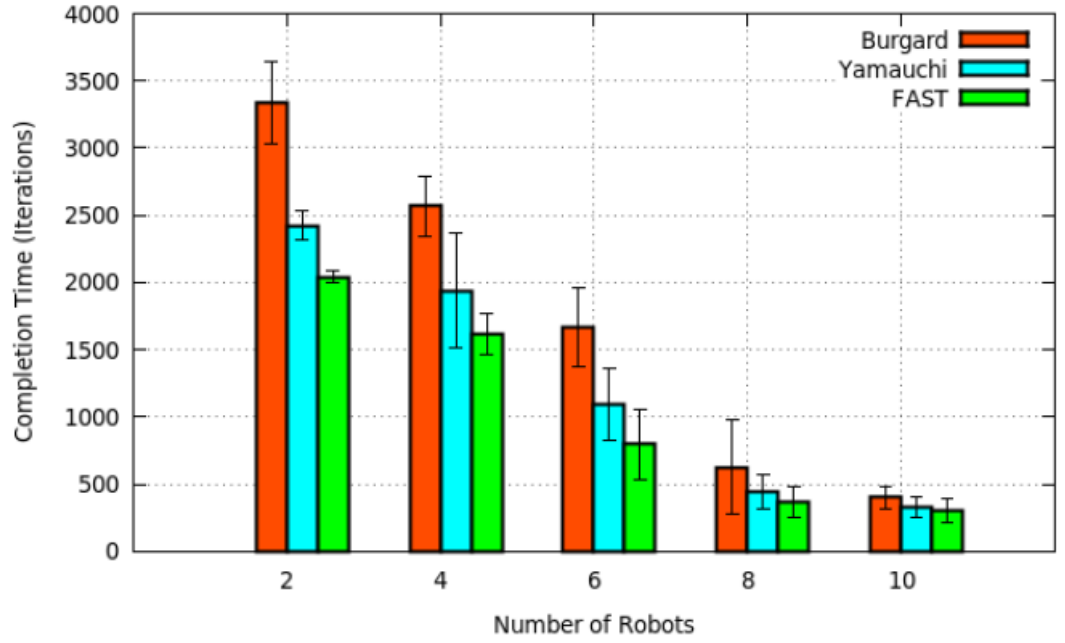


Figure 5.15 Completion time measured in the Outdoor map with one robot failing every 250 iterations (until only one robot is alive)

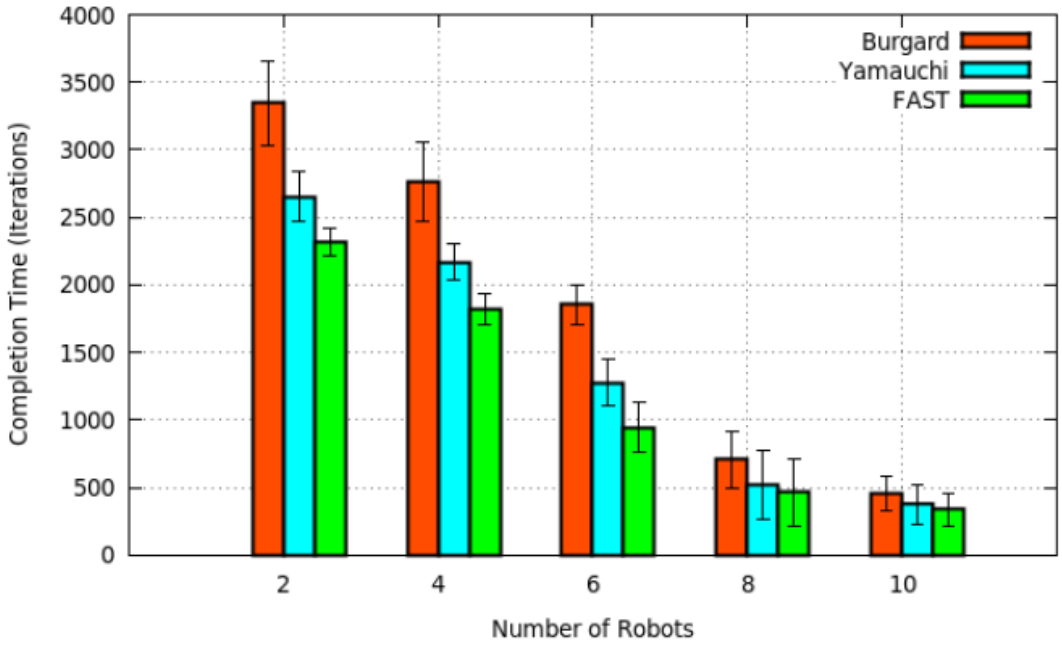


Figure 5.16 Completion time measured in the Office map with one robot failing every 250 iterations (until only one robot is alive)

5.9.2 Physical Experiments

The proposed approach has also been tested on a multi-robot system comprising of Firebird V robots [Firebird 2016]. The robots use an ATMEGA2560 AVR microcontroller and are equipped with an on board 2.4 GHz XBee wireless module with communication rate up to 115.2 Kbps. The XBee modules could be set at 16 different frequency separated channels. On each channel, simultaneous communications without interference is possible. For localization of the robots, a visual marker tracking system was used to uniquely identify each robot and provide it with its global position on the terrain. In all the above experiments, the values of $\tau 1$ and $\tau 2$ are determined experimentally, to suit the Firebird V robots. To determine the value of $\tau 1$, two experiments were conducted:

(a) In the first experiment, a token was passed on a network of 5 robots, for 50 iterations. It was found that reading the token, updating the local map and frontier selection takes 0.589 seconds in the worst case (profiling is done for Algo. 5.2, line 14-17). Therefore, each robot, upon receiving a token, held it for 0.654 seconds (also accounting for packet transmission delay of 65.84 milliseconds) and then transmitted the token to the next robot in the ring. The time for a token to come back to the robot that has generated the token was noted as t_1 and is 3.27 seconds.

(b) In the second experiment, a single robot was made to move randomly from its current location to one of its neighboring cells for 50 iterations. The average time taken by the robot to make a move was noted as t_2 and is no more than 6 seconds in the worst case. From these two experiments, τ_1 was estimated as follows:

$$\tau_1 = \lceil t_1 + t_2 \rceil \tag{5.4}$$

Estimating τ_2 is a two-step process which is described below:

(a) First of all, the packet delay in the XBee network is measured. One robot is programmed to send a message *m* of size 10 bytes to the coordinator of the XBee network (a module attached to a desktop computer) for 50 iterations spaced at an interval of 5 seconds. The robot then waits for the response from the coordinator i.e., an acknowledgement packet. Therefore, the delay is defined in terms of the RTT. In case the ACK is not received by the robot from the coordinator, retransmission is done. On our multi-robot test-bed, the XBee module is connected to the ATMEGA 2560 onboard the robot. This device sends the message *m* to the coordinator of the XBee network and waits for the ACK. It was observed that the delay was between 64 milliseconds (lower bound) and 67 milliseconds (upper bound). The average of 50 iterations (say t_3) was found to be 65.84 milliseconds.

(b) The network setup time in the presence of robot(s) failure is computed. The virtual token ring network of five Firebird V robots is established afresh in 50 iterations. In each iteration one or more robot fails with a locally controlled probability and the network is reconfigured. Here failure of a robot in a particular iteration means that the robot voluntarily stops responding to any message received in that iteration. The reconfiguration process is in accordance with Algorithm 5.4 and Algorithm 5.5, suggested for handling failure. It was found that the multi-robot system is able to gracefully recover from failures in time less than 200 milliseconds (upper bound) in all the iterations. Therefore, network setup time (say t_4) is set to 200 milliseconds.

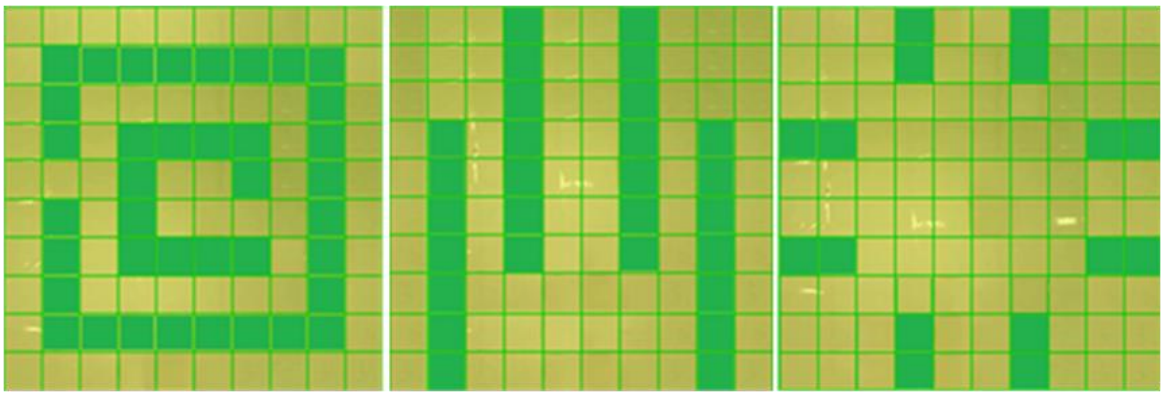
From these two experiments, τ_2 was estimated as follows:

$$\tau_2 = \lceil t_3 + t_4 \rceil \tag{5.5}$$

In order to validate the simulations and also to demonstrate the feasibility of the proposed approach, physical experiments were conducted by considering the multi-robot team size of 2, 3, 4, and 5 robots on three different terrains, as shown in Figure. 5.17. The maps are

discretized into 10×10 square grid cells of size equal to the footprint of the robot which is 0.2 m^2. Two performance metrics have been considered for evaluation of the considered approaches:

- *Completion time* - is measured in seconds and is calculated by starting a timer as soon as the multi-robot system is activated to perform coverage. The start time is recorded as t_1. The timer stops as soon as the last robot in the multi-robot system finishes its coverage task. The end time is recorded as t_2. The completion time is the difference of end time and the start time i.e., t_2-t_1.
- *Percentage redundancy* - is measured in the same way as measured in simulation (refer to Section 5.9.1).



(a) Circular Maze Map (b) Bar Maze Map (c) Living Room Map

Figure 5.17 Three different maps used in terrain coverage experiment

Figures 5.18, 5.20, and 5.22, show the completion time (in seconds) of CAC, BSA-CM, BSA-CMI, and FAST on physical robots for three different maps, viz. Circular Maze, Bar Maze, and Living Room respectively. Irrespective of the terrain map and the coverage algorithm, increasing the number of robots results in reduced completion time. The proposed algorithm (FAST) performs better than all the other state-of-the-art approaches. In BSA-CM and BSA-CMI, the robots select a nearest backtracking point in an iterative manner and covers them first and as a result it takes the two algorithms more time to complete the coverage task.

The performance of a coverage approach is also dependent on overlapping coverage generated by the robots. Overlap in the coverage results in suboptimal utilization of the system resources. Referring to the redundant coverage graph of Circular Maze map, Bar Maze map and Living Room map as shown in Figure 5.19, 5.21 and 5.23 respectively, it can be observed that increasing the number of robots results in increased redundant coverage. In

spite of that, for the proposed algorithm (FAST) the increase in redundant coverage is the least when compared with other approaches. The exact percentage improvement of FAST over other algorithms (in terms of completion time and non-overlapping coverage) is given in Tables 5.4, 5.5, and 5.6 for Circular Maze, Bar Maze, and Living Room respectively.

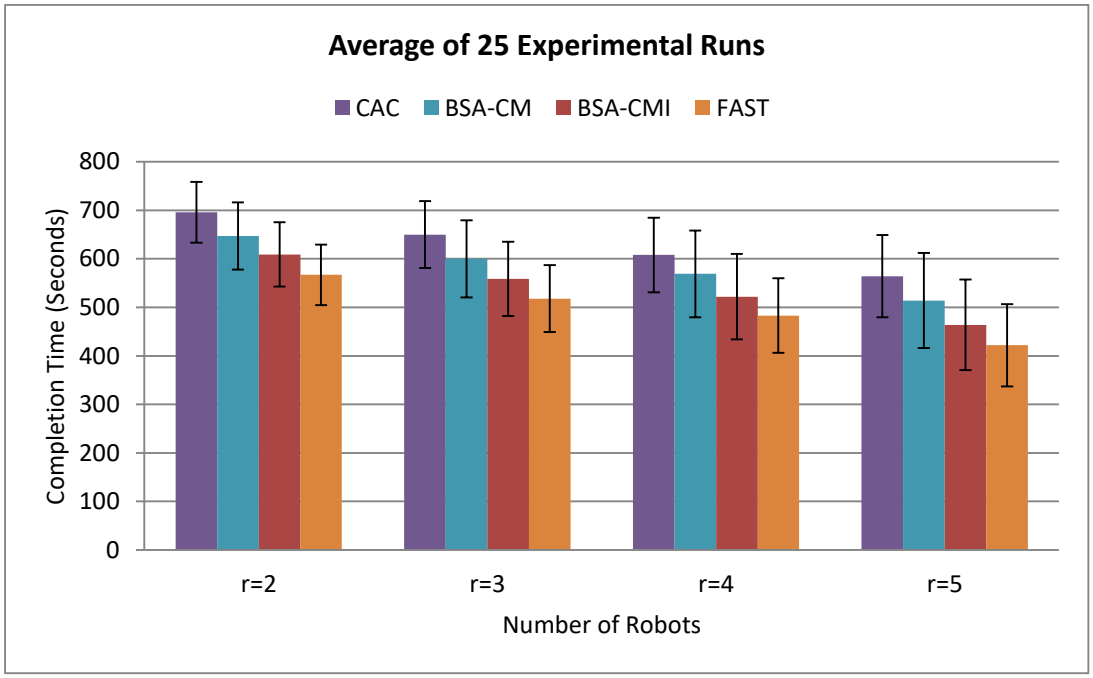


Figure 5.18 Completion Time measured in the Circular Maze Map

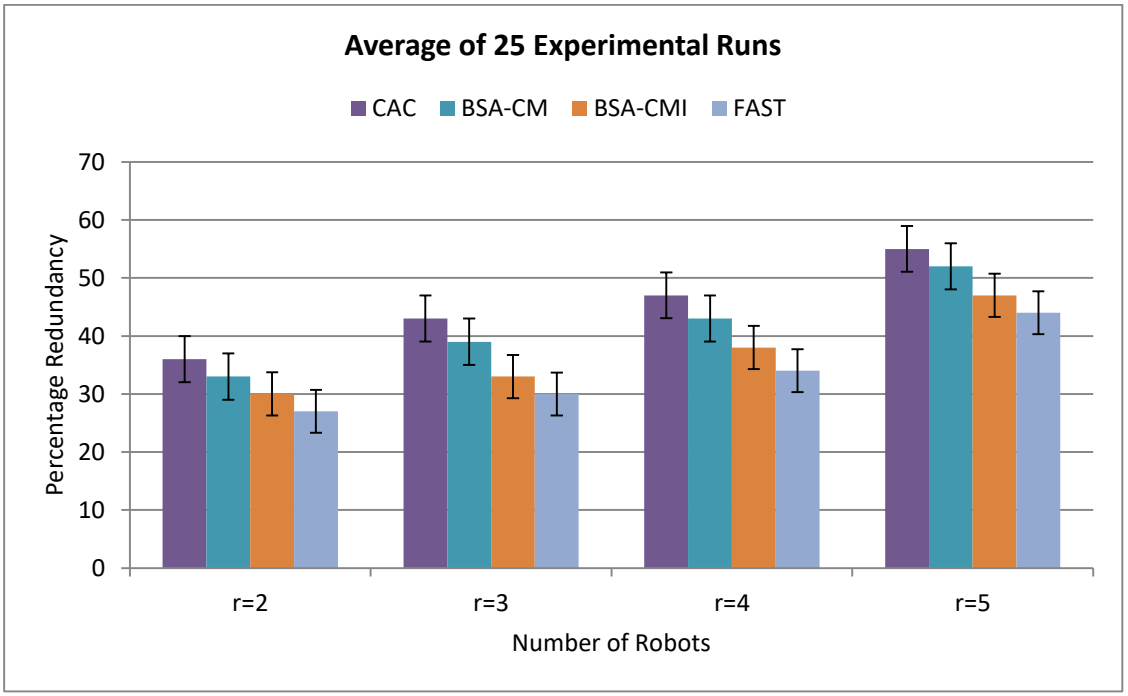


Figure 5.19 Percentage Redundancy measured in the Circular Maze Map

Table 5.4 Comparison of FAST with different approaches in Circular Maze Map

MAP	ROBOTS	FAST vs.	PERCENTAGE IMPROVEMENT IN COVERAGE COMPLETION TIME	PERCENTAGE IMPROVEMENT IN NON-OVERLAPPING COVERAGE
CIRCULAR MAZE	2	CAC	18.5	25
		BSA-CM	12.3	18.18
		BSA-CMI	6.89	10
	3	CAC	20.3	30.2
		BSA-CM	13.66	23.07
		BSA-CMI	7.33	9.09
	4	CAC	20.6	27.7
		BSA-CM	15.11	20.93
		BSA-CMI	7.47	10.52
	5	CAC	25.2	20
		BSA-CM	17.89	15.38
		BSA-CMI	9.05	6.38

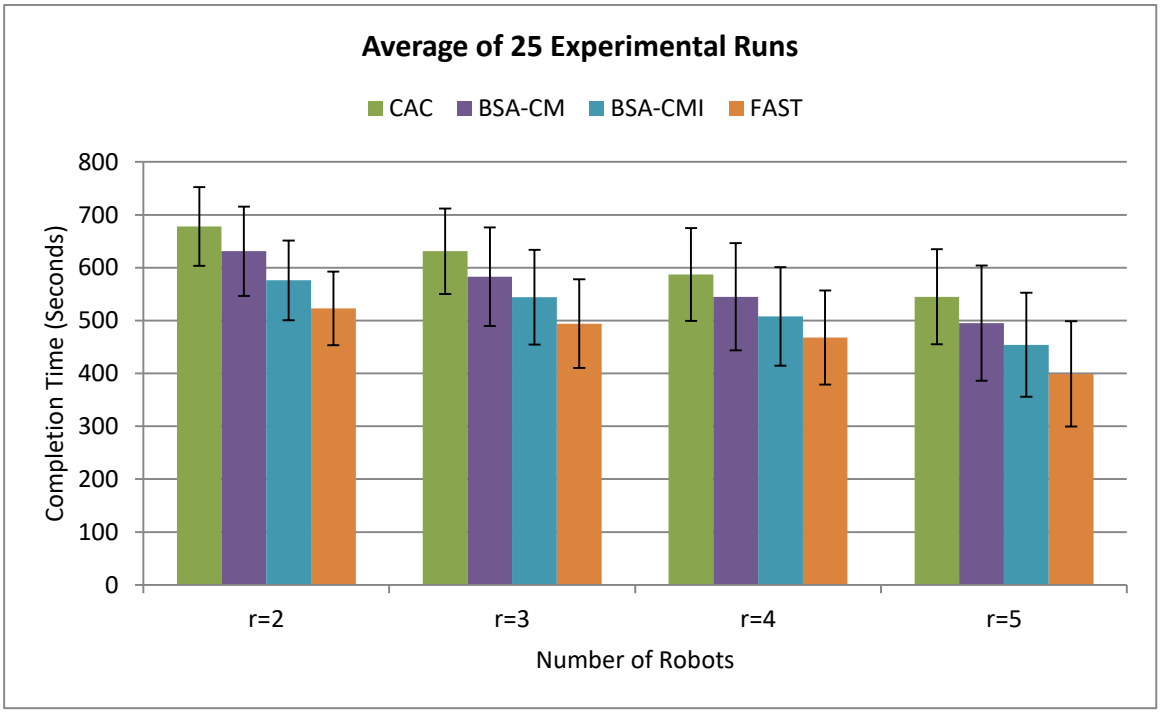


Figure 5.20 Completion Time measured in the Bar Maze Map

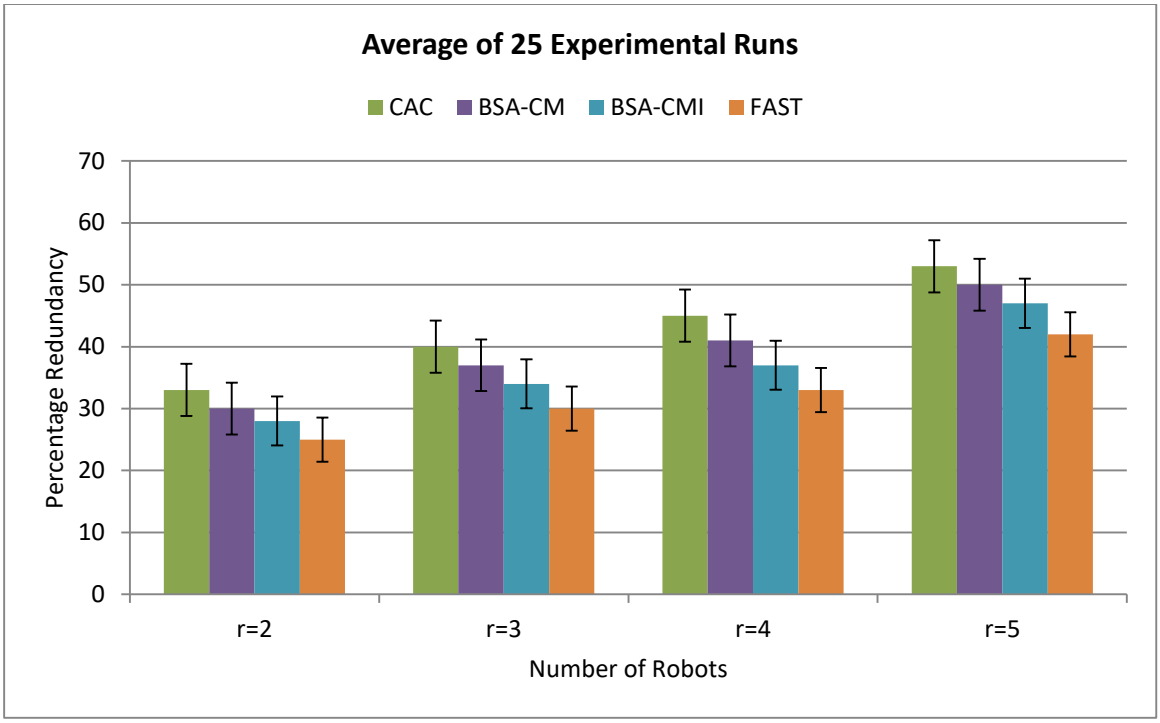


Figure 5.21 Percentage Redundancy measured in the Bar Maze Map

Table 5.5 Comparison of FAST with different approaches in Bar Maze Map

MAP	ROBOTS	FAST vs.	PERCENTAGE IMPROVEMENT IN COVERAGE COMPLETION TIME	PERCENTAGE IMPROVEMENT IN NON-OVERLAPPING COVERAGE
BAR MAZE	2	CAC	22.9	24.2
		BSA-CM	17.11	16.66
		BSA-CMI	9.2	10.71
	3	CAC	21.7	25
		BSA-CM	15.26	18.91
		BSA-CMI	9.19	11.76
	4	CAC	20.3	26.7
		BSA-CM	14.12	19.5
		BSA-CMI	7.87	10.81
	5	CAC	26.8	20.8
		BSA-CM	19.39	16
		BSA-CMI	12.11	10.63

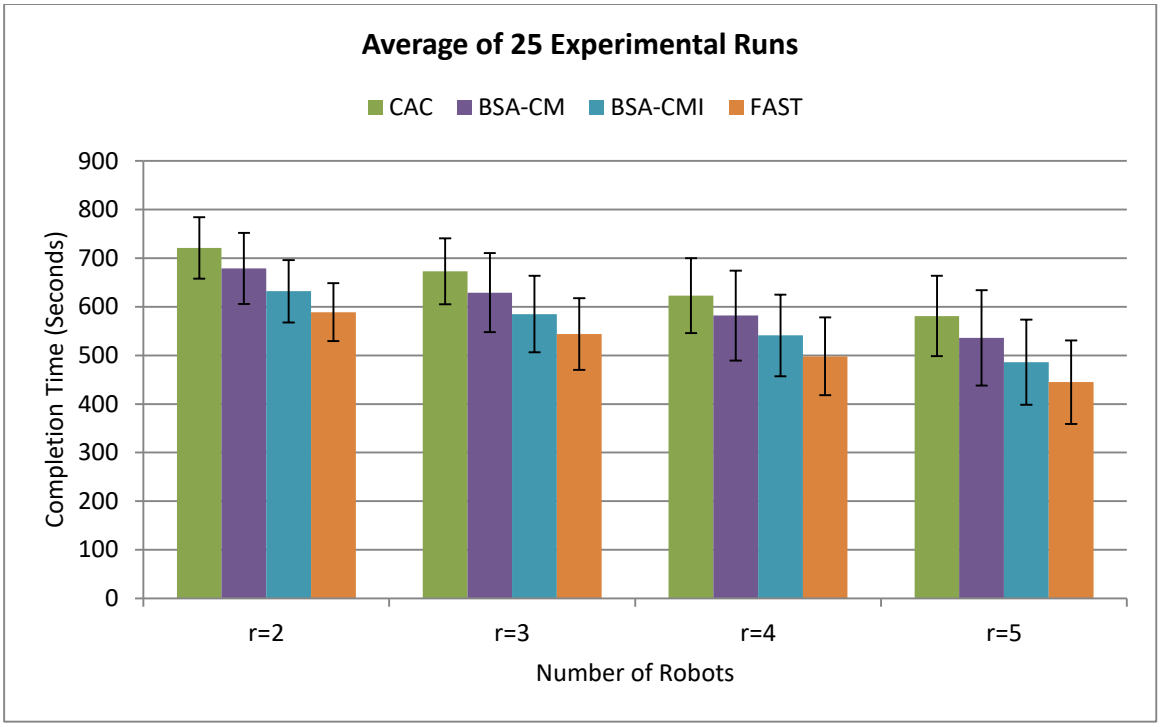


Figure 5.22 Completion Time measured in the Living Room Map

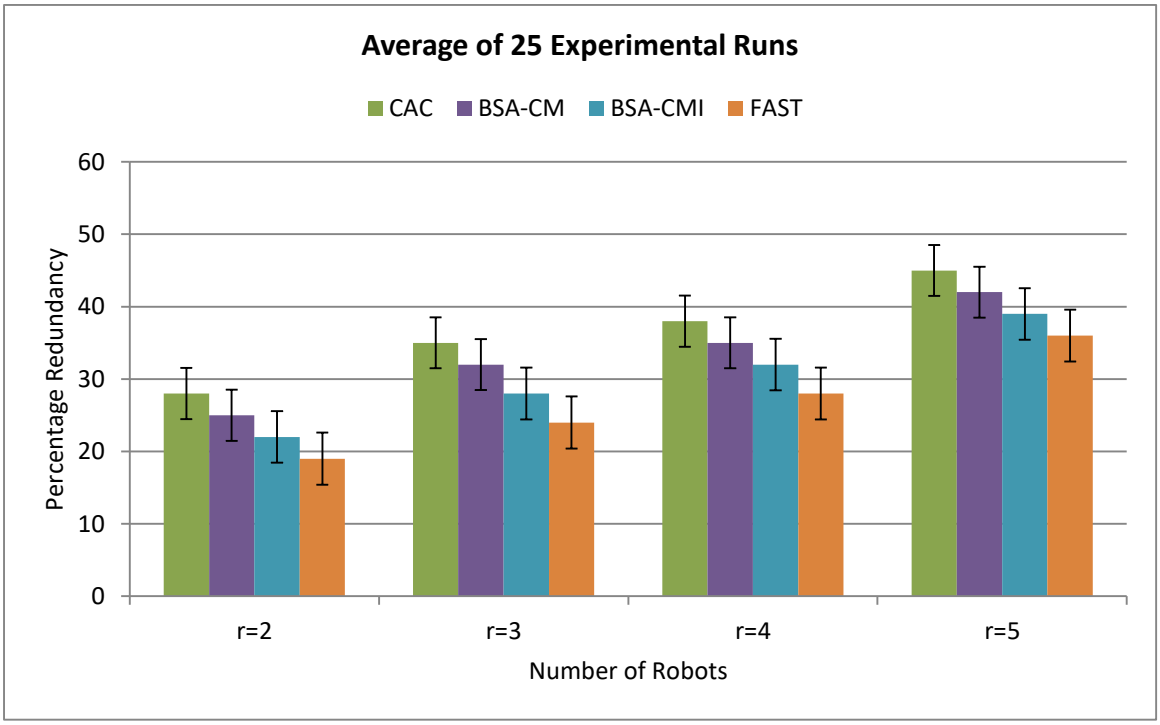


Figure 5.23 Percentage Redundancy measured in the Living Room Map

Table 5.6 Comparison of FAST with different approaches in Living Room Map

MAP	ROBOTS	FAST vs.	PERCENTAGE IMPROVEMENT IN COVERAGE COMPLETION TIME	PERCENTAGE IMPROVEMENT IN NON-OVERLAPPING COVERAGE
LIVING ROOM	2	CAC	18.3	32.14
		BSA-CM	13.25	24
		BSA-CMI	6.8	13.63
	3	CAC	19.16	31.42
		BSA-CM	13.51	25
		BSA-CMI	7	14.28
	4	CAC	20.4	26.31
		BSA-CM	14.34	20
		BSA-CMI	7.94	12.5
	5	CAC	23.8	20
		BSA-CM	16.9	14.28
		BSA-CMI	8.43	7.69

Snapshots of the Firebird V robots executing the proposed algorithm at different stages of execution can be seen in Figure 5.24, 5.25, and 5.26. The map of the environment is not known initially. The status of the white colored cells is not known to the robots. Rather, the robots use their range sensors to detect the status of the eight cells surrounding their current position. In Figure 5.24, three robots are shown to perform coverage of an environment free of obstructions. The preferred direction is set to the north edge of the map. As the robots proceed with the coverage task they generate structured trajectories which results in contiguous coverage of the unknown region. In our experiment as the robots incrementally traverse the frontier cells those cells are marked as covered and are shown in light yellow color. The trajectories of the three robots are shown in three different colors, i.e. blue, purple, and red. In Figure 5.25, three robots are shown to perform coverage in an environment with two randomly placed obstacles. Again, it can be seen that the robots complete the coverage task by contiguously covering all the cells. A similar situation is shown in Figure 5.26 wherein three robots are dispatched to cover the terrain with two obstacles randomly placed in the environment, but this time except one, the other two robots are programmed to fail (stop) after every six iterations. Figure 5.26(c) shows the failure of one robots and Figure 5.26(g) shows the failure of the second robot. In spite of the failure of

two robots one robot (with the blue trail) still completes the terrain coverage task. Also, it can be seen that the terrain coverage pattern is still contiguous.

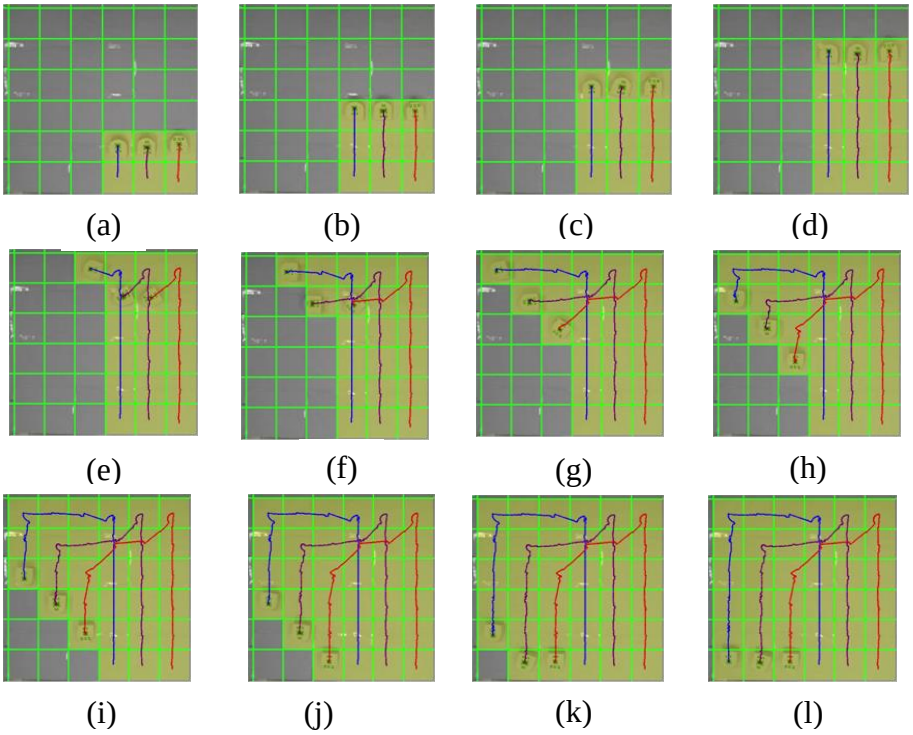


Figure 5.24 Three Firebird V Robots Executing the Algorithm FAST in a Free Space

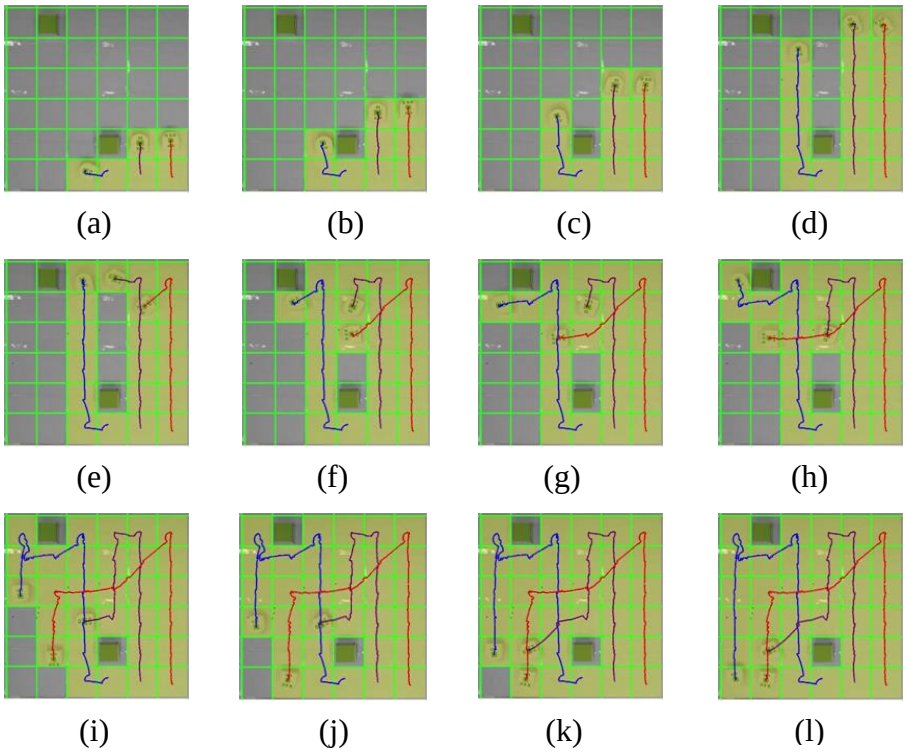


Figure 5.25 Three Firebird V Robots Executing the Algorithm FAST in an Obstructed Environment

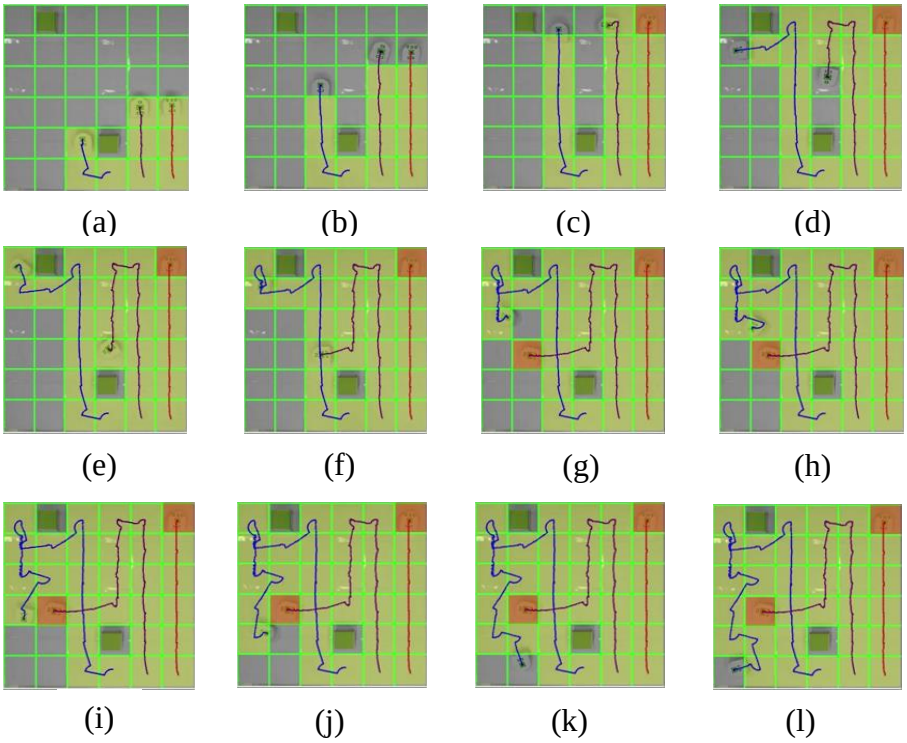


Figure 5.26 Three Firebird V Robots Executing the Algorithm FAST in an Obstructed Environment. One Robot Fails after Every Six Steps

5.10 CHAPTER SUMMARY

A completely distributed approach, **F**rontier **A**llocation **S**ynchronized by **T**oken **P**assing (FAST), is proposed in this chapter. It is a frontier propagation approach wherein the robots use token passing for mutually exclusive selection of the frontier cells. FAST is capable of handling failures of multiple robots. Prior approaches cover the terrain in an irregular fashion without considering the usability of already covered regions. FAST on the other hand, enforces the robots to follow a structured trajectory which is proven to be a powerful approach for coverage path planning method in the literature. The robots have a common notion of a global preferred direction, such that, a robot always selects and moves to a frontier cell in that direction (if an unvisited frontier cell exists). This renders large portions of the terrain usable even before the completion of the coverage task. An information dissemination strategy for map data fusion among the robots is suggested that makes FAST scalable on larger terrains and team sizes. Also, a multi-robot extension of an existing structured trajectory approaches BSA-CM [Gerlein 2011], referred to as BSA-CMI, and is proposed in this paper. The performance of FAST is compared with four different approaches which are representative of the state of the art both in simulation and on a real

multi-robot test-bed. The empirical results thus obtained substantiate the fact that the performance of FAST is better than other approaches.

The team of mobile robots executing the proposed algorithm (FAST) exhibits coordination by way of message passing for the purpose map building, task allocation (frontier selection) and handling failure. Therefore, FAST effectively coordinates the robots by carefully synchronizing the robots in their critical sections. As a result, the algorithm FAST is more efficient when compared with the other state-of-the-art approaches. One important aspect of coordination in a multi-robot system is load balanced distribution of the workload which has been paid least attention. In the next chapter, the problem of decomposition of the unknown region is considered. A multi-robot coordination algorithm is proposed which creates balanced partitions of the unknown region to be processed by each robot.

CHAPTER 6

BALANCED PARTITIONING OF AN UNKNOWN REGION FOR EFFICIENT MULTI-ROBOT COORDINATION

6.1 INTRODUCTION

Autonomous navigation of a mobile robot in an unknown environment is one of the most fundamental tasks. It is evident from the literature that the use of multiple mobile robots increases the efficiency of the task by speeding up the task completion time while at the same time it increases the robustness of the system and makes it fault tolerant [Guzzoni 1997]. The redundancy offered by multi-robot system is often taken advantage of to compensate for sensor uncertainties when overlapping sensor data of different robots is fused to generate a common consistent representation of the world [Juliá 2012]. On the contrary researchers have also studied the risks of interference of active sensor like, ultrasound, IR, laser [Goldberg 1997], [Schneider-Fontán 1998], [Boada 1998] in a multi-robot scenario. Increasing the number of robots only aggravates the problem and robots spend a lot of time in avoiding collisions. Another challenging problem faced when coordinating a group of mobile robots to achieve some common objective is to decompose a bigger task into smaller task subsets and then assigning these task subsets to individual robots in such a way that all robots finish their tasks almost at the same time. In other words, we can say that the multi-robot system should be harnessed to its true potential by load balancing. For instance, consider the task of cleaning a floor in an office building on a regular basis. It is truly advantageous if the multi-robot system only requires a floor plan to execute its mission. The multi-robot system should autonomously segment this floor plan into elementary units or simple regions, like, rooms, halls, and corridors. Further, these elementary units should be combined to create nearly balanced partitions (i.e., partitions of the map that are almost equal in size in terms of area). This goal can only be achieved efficiently if the operational domain of the robot is properly segmented. To this end, there are two commonly used methods to decompose the unknown region into smaller regions. The first method is grid based decomposition [Elfes 1989] and the second method is polygon/ topological decomposition [Wu 2007]. The later method is shown to be more efficient than [Elfes 1989] when it comes to reduced task completion time and reduced sensor interference between the robots [Wurm 2008]. In fact, grid based maps

are not appropriate for navigation and path planning since they have greater reliance on the position of the robots. Also it is difficult to manage and maintain grid based maps especially in bigger regions due to their large memory and computational demands. On the other hand, topological maps are simple and inherently suitable for robot navigation and path planning. Furthermore, the produced maps are translated into graph data structure in which the vertices correspond to the regions like, rooms, halls, and corridors, and edges represent the connectivity of these regions with each other. The topological maps allow application of several algorithms used in graph theory for efficient planning by decoupling the planning algorithm from the absolute and accurate positions of the robots [Portugal 2013]. Richard et. al. [Bormann 2016] has conducted an excellent survey, and have given empirical comparison of four different approaches, that are, *Voronoi graph based segmentation, morphological segmentation, distance transform based algorithm and feature-based partitioning*. Given the floor plan these four approaches are commonly used for room segmentation. The authors have shown that *Voronoi graph based segmentation* method is best both qualitatively and quantitatively because of its compact clusters. This is the primary reason why several researchers have relied on the said method in different applications of multi-robot systems including but not limited to terrain coverage [Breitenmoser 2010], area exploration [Haumann 2010], patrolling [Fazli 2013], and search and rescue [Fu 2009]. But, Voronoi graphs do not produce balanced partitions and therefore many approaches using them are incapable of producing even assignments of the task subsets to individual robots. As a result, some robots have to process large portions of the unknown region while others stay idle. This underutilization of the resources and imbalance of workload in multi-robot system opens up opportunities for further research.

In this chapter, we suggest an approach for creating balanced partitions of an unknown region which are then assigned to multiple robots so that each robot can process their assigned regions in a mutually exclusive manner without interference. The proposed method requires the robots to use minimalistic sensors to first determine the skeleton of the unknown region which is then converted into a weighted connected graph. Further, this graph is partitioned into sub-graphs that are maximally balanced using genetic algorithm. These sub-graphs are optimally assigned to the available robots using an optimal assignment method. To the best of our knowledge this is the first attempt in this direction. The rest of this chapter is organized into eight sections. A discussion on approaches based on polygonal partitioning of the unknown region is presented in Section 6.2. Primary motivation for this

research is presented in Section 6.3. The contributions of this research are presented in Section 6.4. Problem statement and assumptions are detailed in Section 6.5. Section 6.6 provides a detailed description of the proposed approach. In Section 6.7, the problem of maximally balanced connected partitioning of the topological graph using genetic algorithm is discussed. The simulation environment and results are presented in Section 6.8. Chapter summary is presented in Section 6.9.

6.2 A DISCUSSION ON REPRESENTATIVE APPROACHES

In this section, three different region segmentation based approaches for online exploration and coverage are discussed.

6.2.1 Clustering Unknown Space using K-means for Multi-Robot Exploration

Solanas and Garcia [Solanas 2004] have suggested an algorithm for multi-robot exploration in an unknown environment. The bounds of environment are known and no communication restrictions are imposed on the robots. The environment is represented as an occupancy grid map [Elfes 1989]. The cells in the unknown region are clustered using the *K*-means algorithm to obtain as many partitions as the number of robots. These partitions are allocated to the robots for further exploration. As any one robot reaches the centroid of its assigned cluster the remaining unknown space is re-clustered and the process continues until the whole environment is explored. In Figure 6.1, eight robots are shown to perform exploration of an unstructured working space populated with scattered obstacles. The black areas are the already-found obstacles and the irregular spots and rectangles in the background are obstacles that have not been found yet. This is a centralized approach which is computationally expensive. Moreover, the multi-robot system conducts uneven exploration.

6.2.2 Voronoi based Space Partitioning for Multi-Robot Exploration

Wu et. al., [Wu 2007] has extended the previously discussed approach [Solanas 2004]. They strongly criticize the use of occupancy grid maps and K-means clustering for region partitioning and suggest the use of polygonal decomposition i.e., the Voronoi diagrams for spatial partitioning of the unknown region. The rest of the algorithm is similar to [Solanas 2004]. In Figure 6.2, an example of partitioning sequence of an unknown region using Voronoi diagrams with eight robots (K=8) is shown. It is not necessary to know the bounds of the environment and no communication restrictions are imposed on the robots. It is also a centralized approach which is shown to perform better than [Solanas 2004] in terms of the time it takes for the multi-robot system to create partitions of the unknown region as shown

in Figure 6.3. This approach does not guarantee balanced partitioning of the region to be explored by multi-robot system.

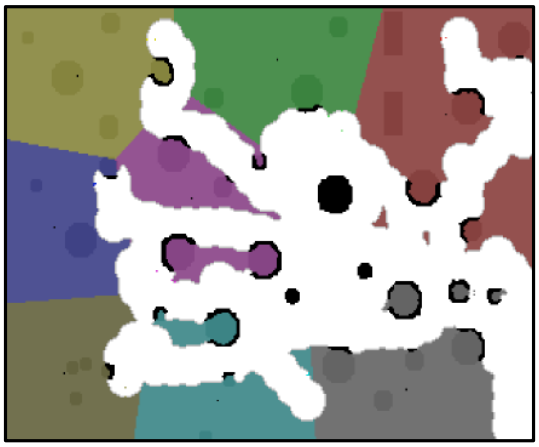


Figure 6.1 K-means clustering of an unknown region suggested in [Solanas 2004]

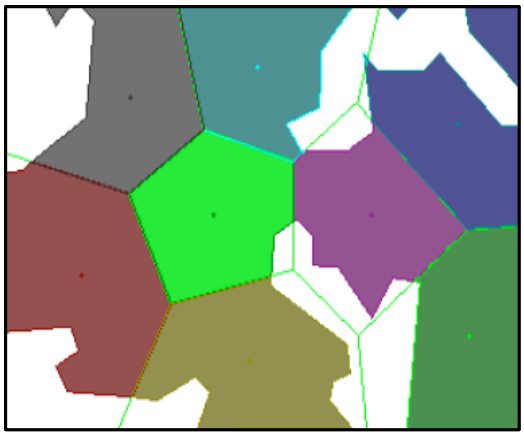


Figure 6.2 Voronoi based spatial clustering of an unknown region suggested in [Wu 2007]

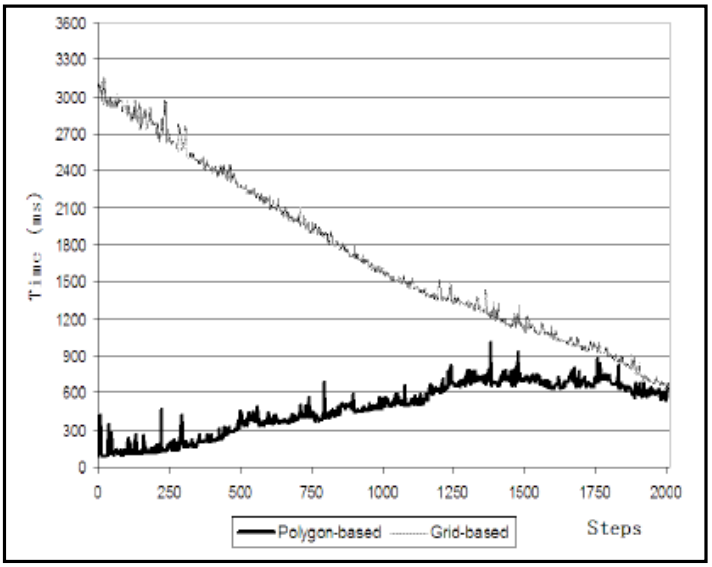


Figure 6.3 Partitioning times during the exploration of a big-size blank map of 400x400 cells with 8 robots [Wu 2007]

6.2.3 Voronoi graph based Segmentation for Multi-Robot Exploration

A Voronoi graph based segmentation method is suggested in [Wurm 2008]. The map of the environment is not known beforehand. However, the bounds of the environment are known. No communication restrictions on the robots are imposed. A Voronoi graph of critical points on the doorways and the narrow spaces is constructed, as shown in Figure 6.4. These critical points are assigned to the robots using an optimal allocation algorithm for further exploration. The suggested approach is compared and shown to be performing better than the basic frontier based exploration algorithm suggested in [Yamauchi 1998]. This approach is a centralized approach and it also does not make any claims regarding balancing the workload of multi-robot system.

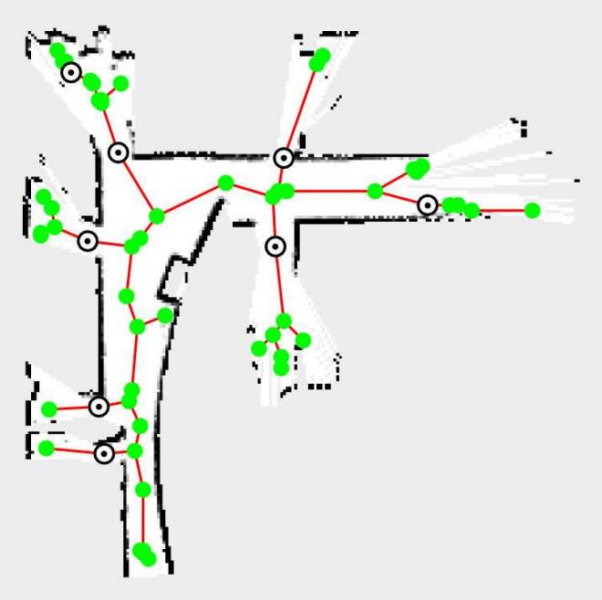


Figure 6.4 Vornoi graph based segmentation of the environment [Wurm 2008]

6.3 MOTIVATION FOR RESEARCH

Our primary motivation for conducting this research comes from the algorithm suggested in [Hungerford 2014, Hungerford 2016] (henceforth referred to as *KH*-Algorithm). The robots are required to perform area coverage and for that they traverse the boundaries of their Voronoi regions. It is shown that some portion of the robot's partition becomes inaccessible due to the presence of obstacles as shown in Figure 6.5(a). The inaccessible patches once identified are auctioned to other robots in the adjoining regions. Initially the robots are sufficiently dispersed in the unknown environment and only after that the Voronoi partitioning is done. The main problem is that the partitions thus created are of uneven sizes.

If the robots are allowed to process the partitions right away it will be an unbalanced utilization of system resources because some robots will have to process larger regions. The problem is further aggravated when some portions of the robot's partition becomes inaccessible due to the presence of obstacles as shown in Figure 6.5(b). The imbalance in workloads of the robots i.e., the size of the region that each robot has to process leaves certain robots idle leading to underutilization of the resources. This factor is not considered in *KH*-Algorithm. The proposed approach extends *KH*-Algorithm for balanced partitioning and allocation of the unknown region to the multi-robot system. Our approach is particularly suitable for indoor environments like office buildings, hospitals, department buildings in universities etc. Such environments comprise of a series of rooms, corridors, waiting halls, lobbies etc. The proposed approach successfully produces balanced spatial partitions in such environments.

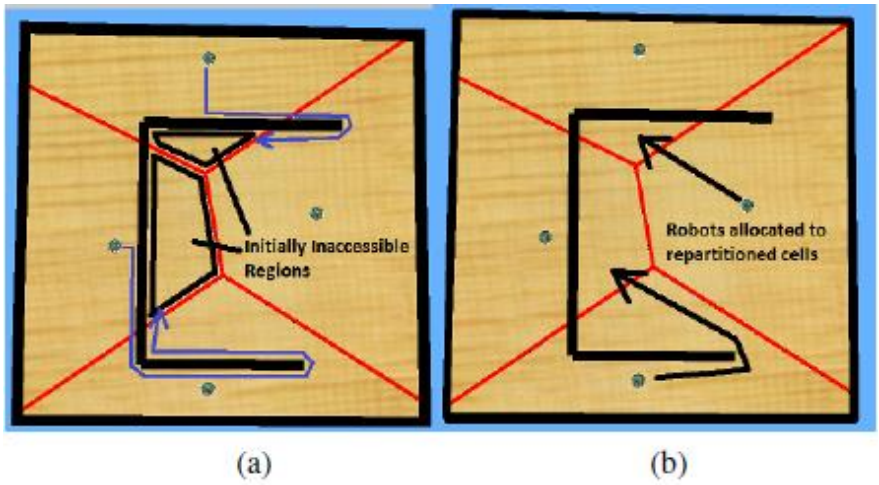


Figure 6.5 (a) Two robots have a portion of their Voronoi partitions inaccessible (b) The inaccessible portions are auctioned to the robots in the adjoining partitions [Hungerford 2016]

6.4 OUR CONTRIBUTION

1. The *KH*-Algorithm is solely based on Voronoi partitioning and auctioning for spatial partitioning of the unknown environment. We have implemented this algorithm and analyzed the imbalance in the spatial partitioning of the unknown environment.
2. We have extended the *KH*-Algorithm by constructing a topological graph on top of the partitions produced, by using the algorithm suggested in [Thrun 1998], which is one of the seminal works in mobile robot navigation.

3. A significant contribution of the proposed approach is that the spatial partitioning process is independent of the initial position of the robots. On the other hand, the size and shapes of the partitions produced by many approaches [Wu 2007], [Wurm 2008], [Solanas 2004], [Fu 2009], and [Fazli 2010] including the *KH*-Algorithm depends on the initial positions of the robots and therefore produce uneven partitions.
4. The topological graph thus obtained is partitioned into maximally balanced and connected partitions using genetic algorithms [Cincotti 2002] which are then assigned to the individual robots using the Hungarian method [Kuhn 1955].
5. This leads to fair division of the workload among the robots i.e., almost balanced partitions in terms of the area that is to be processed by each robot and as a result the multi-robot system is harnessed to its true potential.

6.5 PROBLEM STATEMENT

6.5.1 Robot's Workload Defined

The workload of the robot is defined as the size of the region each robot needs to process for the purpose of cleaning, exploration, search, patrolling and variety of other tasks. The proposed approach aims at partitioning the unknown environment into *n* different partitions (where *n* is the number of robots) and then assigns each region to unique robots. The number of robots to be dispatched completely depends on the capabilities of the robots and the size of the workspace and is decided by the user. Further, each partition of the unknown environment should be of equal size in ideal situations.

6.5.2 Settings and Assumptions

We have assumed that the unknown region is bounded and has polygonal boundary, however, the bounds are not known initially. It is also assumed that the unknown region consists of polygonal obstacles and whole of the free space is a single connected component. The multi-robot team with the following capabilities is considered:

1. All robots are homogeneous and are equipped with 12 short range sensors evenly spaced at 30 degrees (for e.g. Infrared) for obstacle detection.
2. There are no communication constraints and every robot is able to communicate with all other robots within the bounds of the unknown region. This allows the robots to exchange information relating to the task completion status of their assigned regions with other robots.

3. The robots share a common global reference frame and are equipped with sensors to localize themselves in it. This allows the robots to exchange information relating to the skeleton of their assigned regions with their peers.

The detailed description of the proposed approach for balanced partitioning of the unknown region for efficient multi-robot coordination and task completion is discussed in the next section.

6.6 PROPOSED APPROACH

For efficient utilization of the system resources (i.e., the robots) a task must be evenly divided amongst the members of the multi-robot system. The unknown region is decomposed into several smaller partitions that are again intelligently merged to create balanced partitions. The proposed approach is represented with the help of a flow-chart shown in Figure 6.6. Subsequently each processing step of the flow chart is explained separately.

Step1: The first step relates to dispersing the robots in an unknown environment. The proposed approach makes no assumption that the robots are already dispersed inside the unknown region, rather the robots start from a compact initial configuration. It is necessary because the robots are going to partition the unknown environment with respect to a common global reference frame. The dispersion of robots is achieved by using the potential field based approach suggested in [Howard 2002] where it is used for deploying a mobile sensor network in an unknown environment.

Step2: Once the robots are sufficiently dispersed in the unknown environment, Voronoi partitioning is used for decomposing the unknown region into smaller partitions by considering the position of the robots as Voronoi generator.

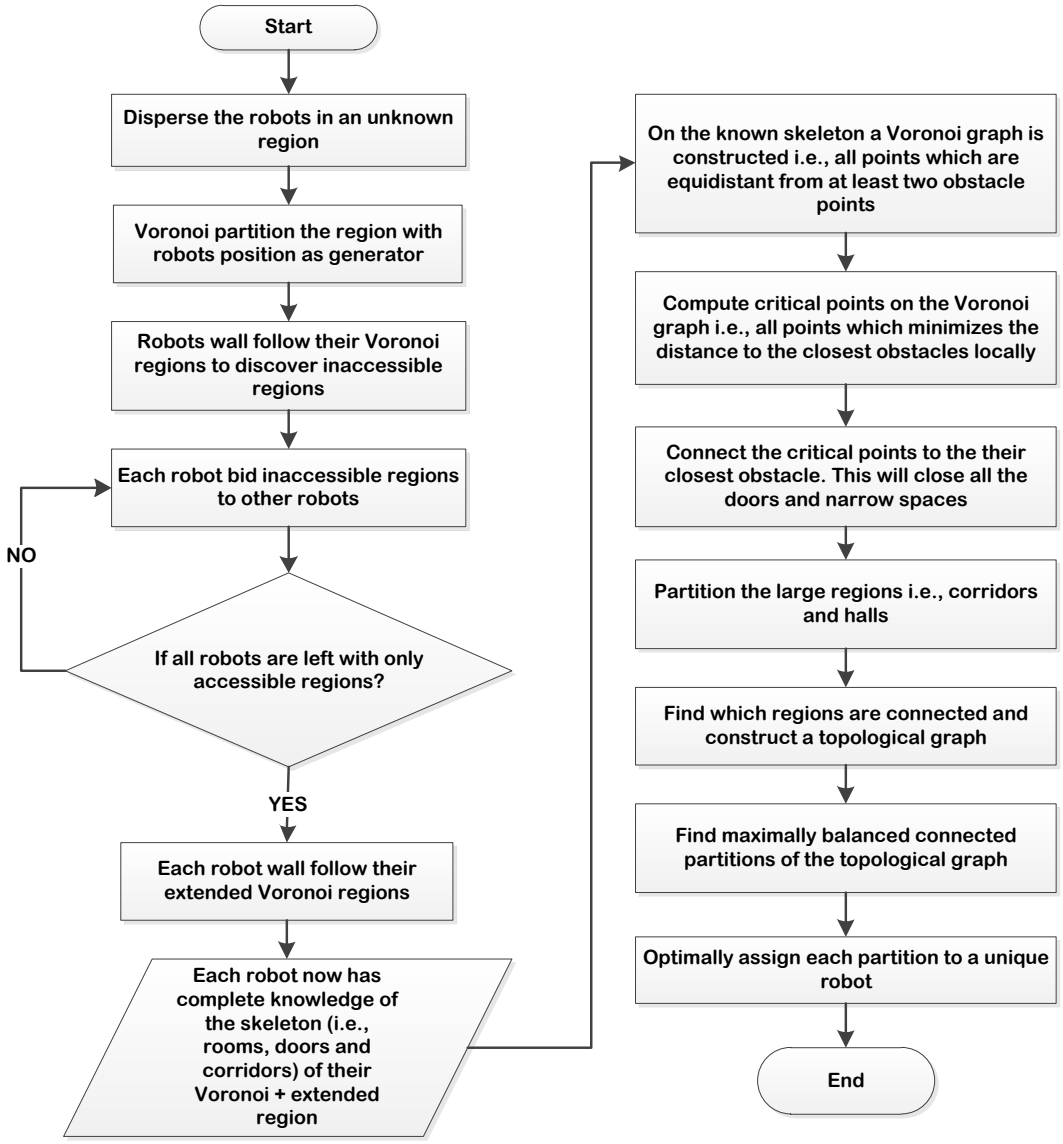


Figure 6.6 Flowchart of the proposed balanced partitioning approach

Step3: The Voronoi partitioning leaves only one robot inside each smaller partition. In computational geometry, this partition is nothing but the region of dominance of each generator (robot in this case) and is referred to as dom(r). Every robot is supposed to process its own partition in a mutually exclusive manner. The partitions of two robots may become inaccessible due to the presence of obstacles as shown in Figure 6.5(a). We start with dispatching each robot to follow wall and boundary of the allotted region to discover the skeleton and the inaccessible portions of their respective partitions (we refer to this as goal-1). At any instance of time, the robots only use their infrared sensors. These sensors are simple with relatively low cost and good for the tasks of obstacle avoidance and wall following in an unknown environment. Moreover, these sensors have faster response time and can easily be

integrated on variety of platforms. For achieving goal-1, Bug-2 algorithm is used. Bug algorithms have proven termination conditions [Choset 2005] and are frequently used for robot navigation in an unknown environment. This algorithm moves the robot on the line joining the robot's present position and the goal (referred to as m-line), unless an obstacle is encountered. If an obstacle is encountered the robot switches to wall following behavior around the obstacle until the motion towards the goal on the m-line is allowed once again. However, one movement restriction is imposed on the robots i.e., the robots should not move outside of their Voronoi partitions. Therefore, the Voronoi region of one robot is considered as obstacle by other robots. The output of this step is shown in Figure 6.7.

Figure 6.7 Boundaries of respective Voronoi partitions as discovered by robots

Step4: At this stage the robots have knowledge of the skeleton of their respective partitions i.e., the accessible and inaccessible portions of their Voronoi partitions. The robots sequentially start auctioning the inaccessible portions of their territories to other robots. All the other robots who can access the inaccessible regions in the current auction respond with a finite bid. The bid value is equal to the size of the area already allocated to the bidder robot. The auctioneer robot then assigns the inaccessible region to the robot with the lowest bid. The process is repeated until all the robots are left with only accessible regions. As a result of this auctioning the territories of some robots is extended. The bounds of the extended territories are mapped again by executing wall following behavior. At the end of this stage, every robot has complete knowledge of the skeleton (the boundaries of different rooms, doors and corridors) of their Voronoi partitions. The output of this step is shown in Figure 6.8.

Step5: This point onwards we have implemented the algorithm for the construction of topological maps suggested in one of the seminal works for indoor mobile robot navigation [Thrun 1998]. The author has introduced the concept of critical lines, representing narrow passages, such as doorways, that are found by analyzing the skeleton of the environment. The free space is denoted by C and the occupied space is denoted by $\bar{C}$. Each point $<x, y> \in C$ has one or more nearest points in the occupied space $\bar{C}$. These points are referred to as basis points of $<x, y>$. The distance between $<x, y>$ and its basis points is referred to as clearance of $<x, y>$. Considering the skeleton of the unknown region derived in the previous step, a Voronoi graph (G_{VG}) is constructed by computing a set of points $<x, y> \in C$ which are equidistant from at least two different basis points as shown in Figure 6.9.

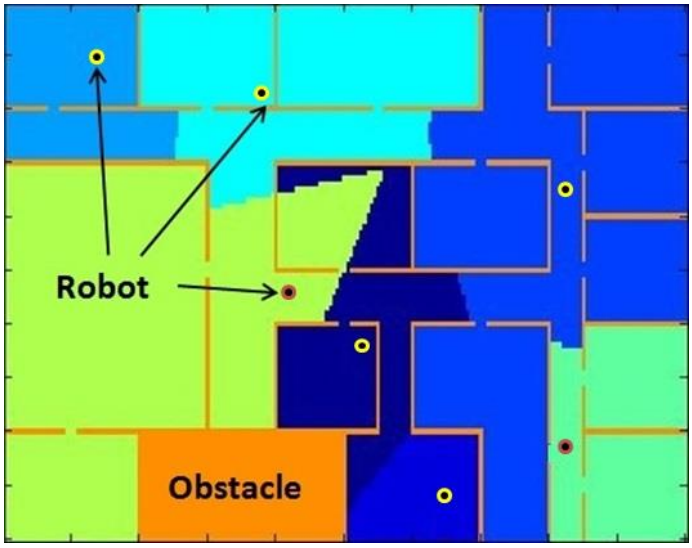


Figure 6.8 Territories of robots after auctioning the inaccessible portions of their respective partitions

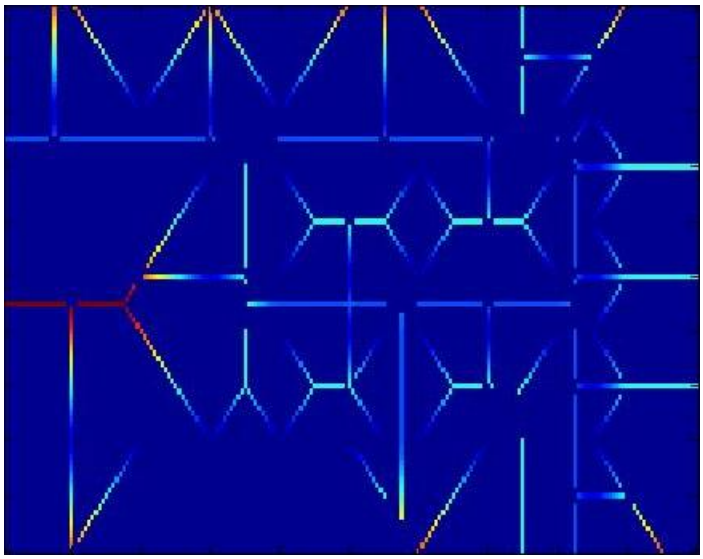


Figure 6.9 Voronoi graph as computed on the skeleton of the unknown environment

Step6: We compute "critical points" for partitioning the free space. Basically, critical points $<x, y> \in G_{VG}$ locally minimize the clearance of $<x, y>$ [Thrun 1998]. The output of this step is shown in Figure 6.10. These points are shown in white color and are mostly discovered at narrow passages and doorways. Some critical points are discovered in succession and are referred to as *Clines*. They are found mostly in the regions representing rooms and corridors.

Step7: The critical points thus computed are connected to their basis points, and the resulting lines are referred to as critical lines [Thrun 1998]. These critical lines virtually close all the doors and narrow spaces as shown in Figure 6.11. This partitions the workspace into simple regions i.e., rooms, halls and corridors.

Step8: It is possible that some of the simple regions (for example, corridors, meeting rooms, waiting rooms, dining halls etc.) may be large in size. They are again partitioned using two different strategies, one for corridors and another for the large rooms. Corridors are detected when critical points constitute more than two consecutive cells in the area; refer to *Clines* as shown in Figure 6.10. These lines of critical points lead to inaccessible regions when the critical points are connected to their basis points. To prevent such situations, only the mid-point of the *Clines* is retained and all the other critical points on this line are deleted as shown in Figure 6.12. It ensures that the corridors are represented by only one critical point resulting in two different partitions of the corridor. Maps with large rooms that can only be allocated to one robot creates imbalance i.e., the area of the rooms is greater than the area that each robot would get, if the total available area is equally divided between the available robots. Therefore, such regions are also required to be decomposed in a manner that no region has an area greater than total area/number of robots. When such regions are detected in the partitioned map, our algorithm finds two points $(2X_l+X_m/3,\ 2Y_l+Y_m/3)$ and $(X_l+2X_m/3,\ Y_l+2Y_m/3)$, where (X_l, Y_l) are the lowest X and Y coordinates respectively and (X_m, Y_m) are the highest X and Y coordinates respectively in the region under consideration. Such regions are then partitioned into two different regions by Voronoi partitioning the area with the two points as generators. This process is repeated till all the regions in the map have areas less than the limit. The output of this stage is shown in Figure 6.13.

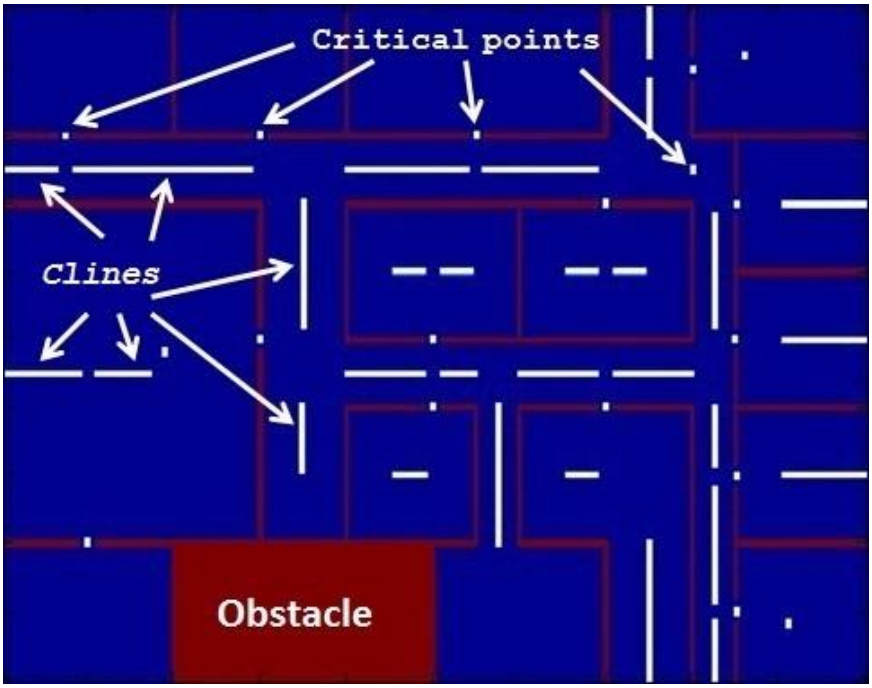


Figure 6.10 Critical points which locally minimize the clearance of some point on Voronoi graph

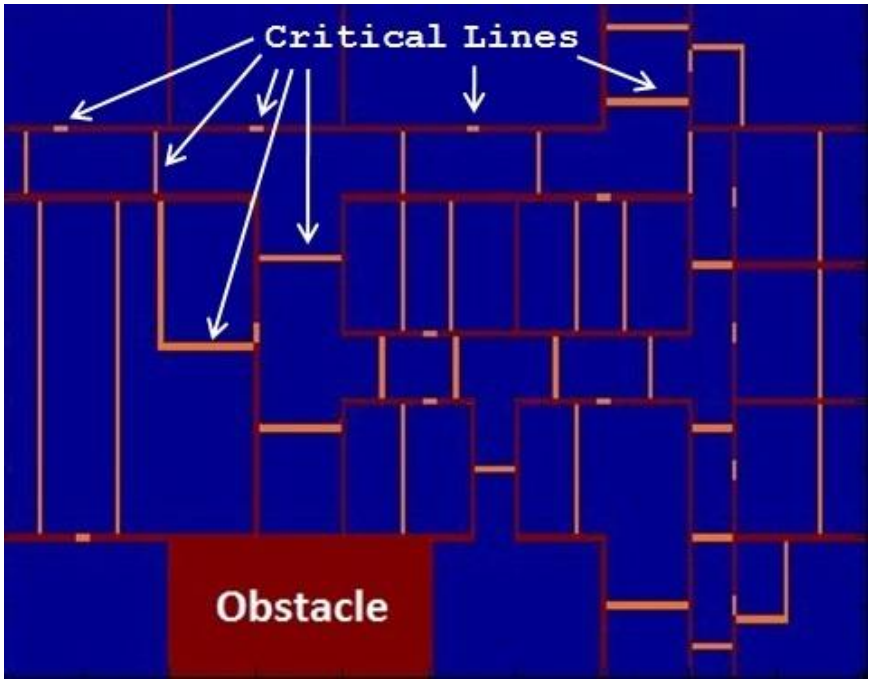


Figure 6.11 Critical points are connected to their basis points

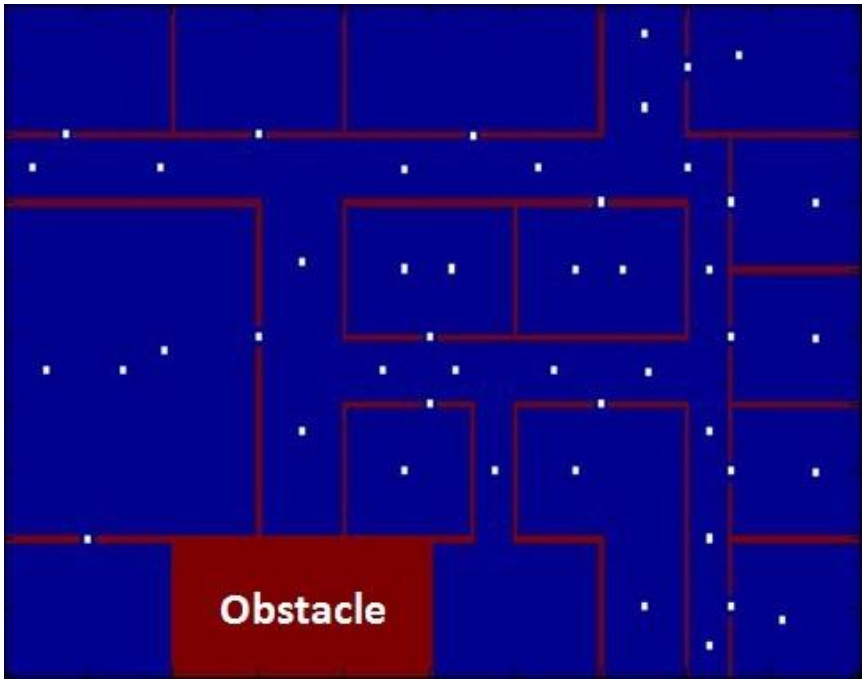


Figure 6.12 All Clines are reduced into a single critical point as shown in white color

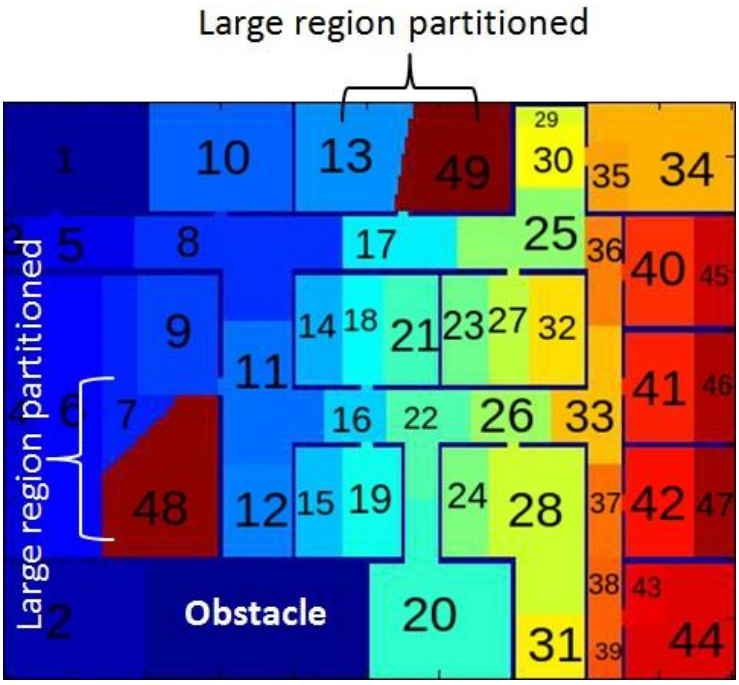


Figure 6.13 The unknown region is partitioned into simple regions labeled from 1 to 44

Step9: Each simple region thus found corresponds to a vertex of a graph. An edge connecting two simple regions corresponds to a critical line separating the two regions. This is motivated by the fact that whenever a robot has to travel from one region to the other region it will have to pass through the narrow passages and the doorways. The output of this step is a topological graph (G_T) which is shown in Figure 6.14.

Step10: The graph (G_T) is required to be partitioned into maximally balanced connected partitions (henceforth referred to as MBCP) on the basis of vertex weights, where the weight of each vertex corresponds to the area of the region that this vertex encompasses. We want to find as many partitions as the number of robots. Finding MBCP is proven to be NP hard [Chlebíková 1996]. For this purpose, we have used genetic algorithms. Further details are provided in Section 6.7. In the process of finding MBCP, many simple regions are merged into one partition. The output of this step i.e., the partitions we have obtained are shown in Figure 6.15. The partitions as shown in Figure 6.13 are merged with respect to the graph partitioning process. The final results are shown in Figure 6.16.

Step11: Once the partitions of the graph are obtained they are assigned to the individual robots using an optimal assignment algorithm i.e., using the Hungarian method. Both centralized [Kuhn 1955] and distributed [Giordani 2010] implementations of the Hungarian method are available.

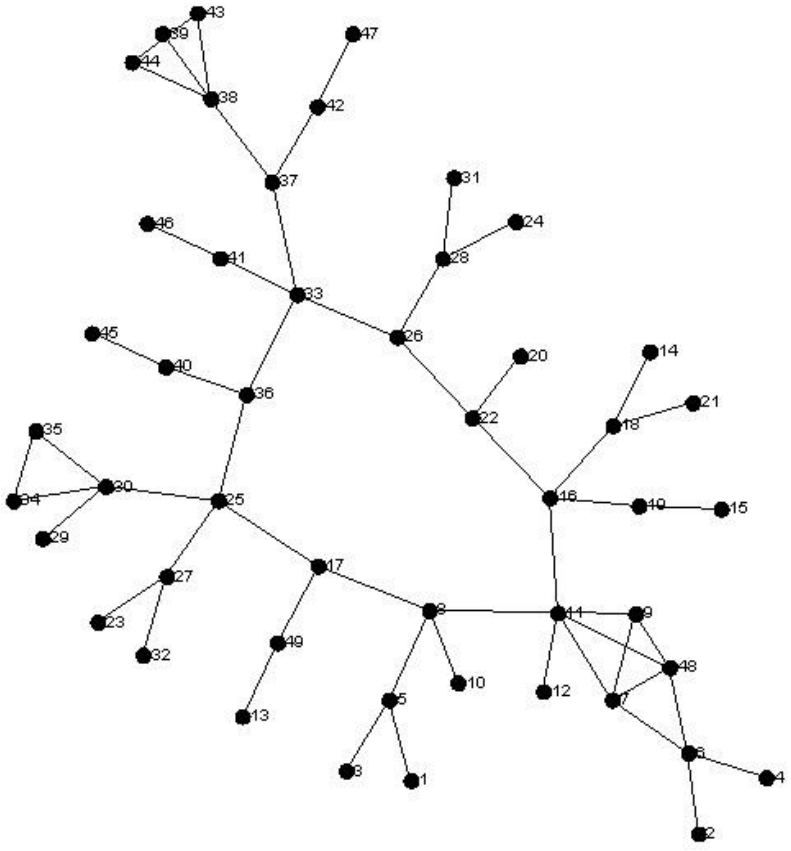


Figure 6.14 The partitioned unknown region as shown in Figure 6.12 is converted into a topological graph (G_T)

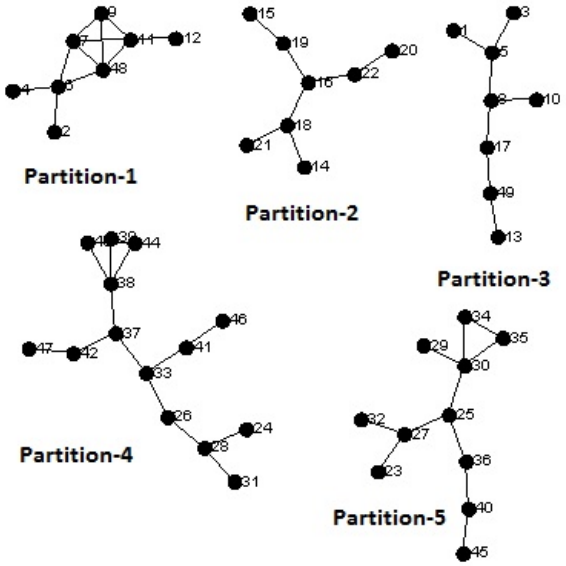


Figure 6.15 The topological graph (G_T) is partitioned into five different maximally balanced partitions for K=5 robots

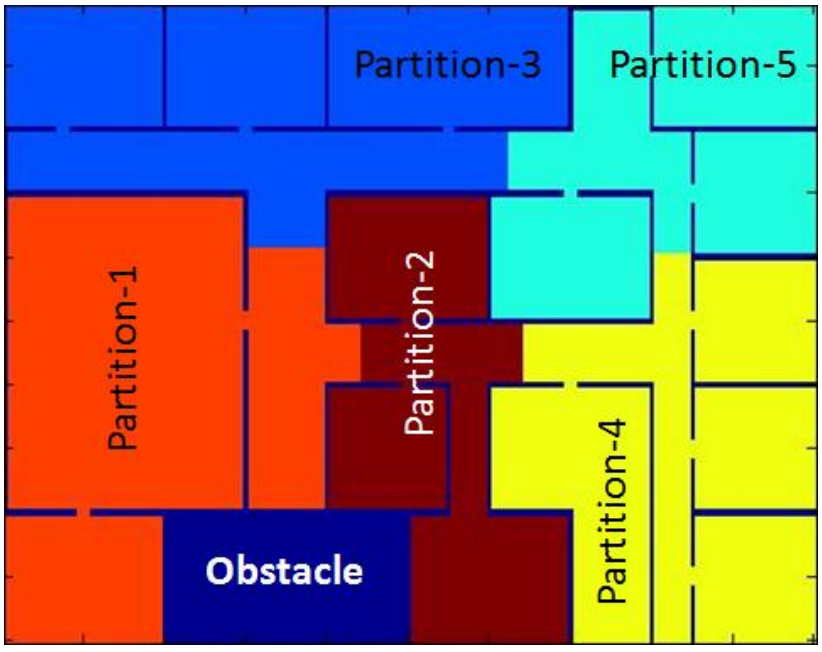


Figure 6.16 The partitioned regions as shown in Figure 6.13 are merged with respect to the partitions of the topological graph as shown in Figure 6.15

The robots can now start processing their respective partitions. We want to stress on the fact that variety of multi-robot tasks can be performed without much interference of the robots with each other. For example, terrain coverage for the purpose of cleaning, mapping and exploration, searching for some objects of interest, patrolling etc.

6.7 MAXIMALLY BALANCED CONNECTED PARTITIONING OF THE TOPOLOGICAL GRAPH

Let us denote the topological graph obtained in *Step 9* of the previous section as $G_T(V, E, W_i)$, such that, $V = \{ V_1, V_2, V_3, .. , V_m \}$ be the number of vertices and $|V| = m$, E be the set of undirected edges connecting the two vertices V_i *and* V_j and W_i be the weight of vertex $V_i \in V$. All the edges have weight equal to one. Here each vertex $V_i \in V$ represents a simple region discussed earlier, and the weight (W_i) of the vertex (V_i) is the area that this vertex encompasses. The maximally balanced connected partitioning (*MBCP*) of the topological graph $G_T (V, E, W_i)$ is to partition the graph into n non-empty partitions (where n is the number of robots) $P = \{ P_1, P_2 ,... , P_n \}$ such that, the sub-graphs induced by each partition $P_i \in P$ is connected and the difference between the sum of the weights of the vertices of each partition is minimum. Formally, the problem is defined in [Cincotti 2002] as graph partitioning problem and is stated as follows:

- $V = \bigcup P_i$ and $P_i \bigcap P_j = \emptyset$ for $i \neq j$
- $W(P_i) \equiv W/n$ where $W(P_i)$ and W are the total sum of the weights of vertices in partition P_i and the total sum of the weights of the vertices in V respectively.
- *Minimum cut objective*: The sum of the weights of the edges crossing between the partitions, called the edge separators, is minimized.
- Let us normalize the vertex W_i weights to sum up to 1: $\sum_{i=1}^{m} W_i = 1$. The balancing factor in accordance with the minimum cut objective is defined as:

$$n.max_{l \in P} \sum_{j=l} W_j \leq t \qquad (6.1)$$

  The left hand side of the inequality in eq. 7.1 is the ratio of the biggest partition in terms of cumulative normalized vertex weight to the desired equal partition size. Perfectly balanced partitions are obtained for $t=1$ but this value is chosen to be slightly higher than 1 i.e., 1.05 to allow 5% imbalance.

We have used the balancing factor in equation 6.1 as a fitness measure of the genetic algorithm which is used for the purpose of graph partitioning and is adopted from [Cincotti 2002].

6.8 RESULTS AND DISCUSSIONS

In this section, the description of the simulation framework is presented in section 6.8.1 followed by simulation results in section 6.8.2.

6.8.1 The Simulation Framework

The proposed approach has been implemented in simulation using Matlab R2015a on an Intel Core-i5 machine with 8 GB of RAM. The resolution of the three maps is set to 0.05 meter per grid cell. The ground area of the three maps is 500 m^2, 750 m^2 and 1000 m^2 for Banquet Hall, Office Map and Hospital Map respectively. The robots are simulated as a point mass equipped with 12 IR based proximity sensors evenly spaced at 30^0 to detect obstacles and walls. It is assumed that the robots are able to accurately localize themselves and their peers. The authors in [Vasisht 2016] have demonstrated accurate localization in decimeter range using a single WiFi access point. It is also assumed that the robots are able to communicate with each other across all positions of the map. The graph partitioning algorithm explained in the previous section is implemented in Java using [GraphStream 2016] which is a graph handling Java library used for the purpose of visualization of the topological graph (G_T) and its partitions (P).

6.8.2 Simulation Results

The proposed algorithm has been evaluated through a series of simulation runs on three different types of indoor environments as shown in Figures 6.17, 6.18 and 6.19. The algorithm is compared with *KH*-Algorithm which is a recent representative of Voronoi based decomposition approaches. Figure 6.17 is a hospital map with rooms and corridors of different size and width. Figure 6.18 is an office map with equal sized rooms and corridor of the same width. Figure 6.19 is a banquet hall with four big rooms each of equal size and a wide corridor.

Since the spatial partitioning process of the proposed approach is independent of the position of the robots the partitions are deterministically computed. Tables 6.1, 6.2 and 6.3 show the results of partitioning the Hospital map, Office map and the Banquet Hall respectively. Since the size of the partitions is strictly dependent on the initial positions of the robots in *KH*-Algorithm, we have conducted a hundred simulation runs with random robot positions and have computed the average results. For the *KH*-Algorithm, Tables 6.4, 6.5 and 6.6 show the average standard deviation, average area of the biggest partitions, and average area of the smallest partitions respectively, for three different maps for different number of robots.

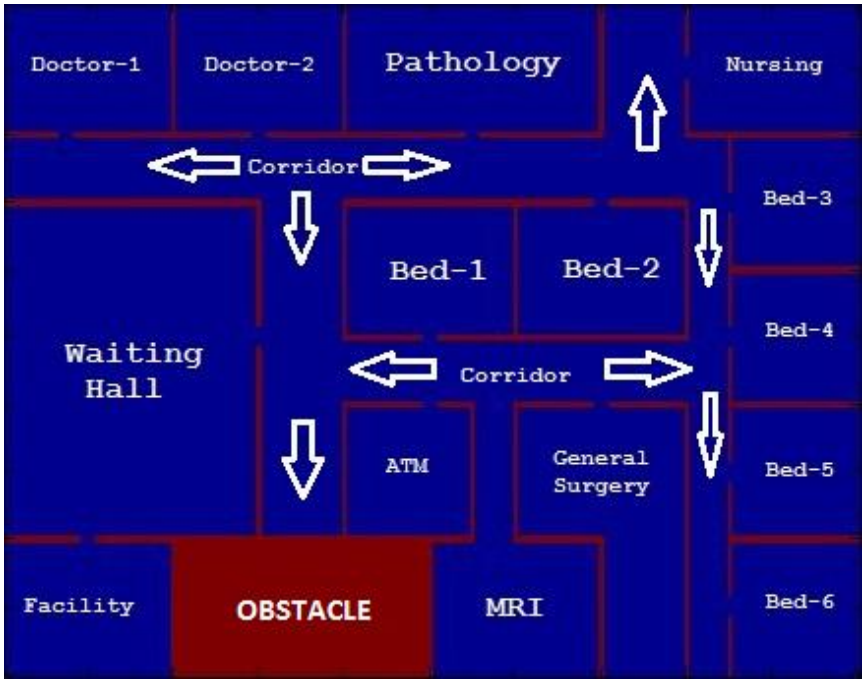


Figure 6.17 Hospital map with rooms and corridors of different size and width

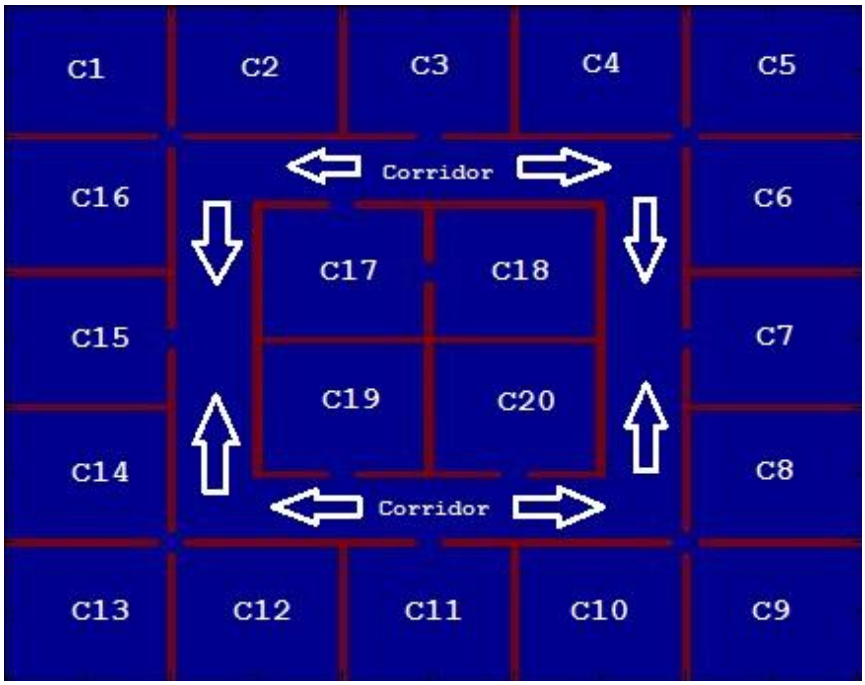


Figure 6.18 Office map with 20 rooms of same size and corridor

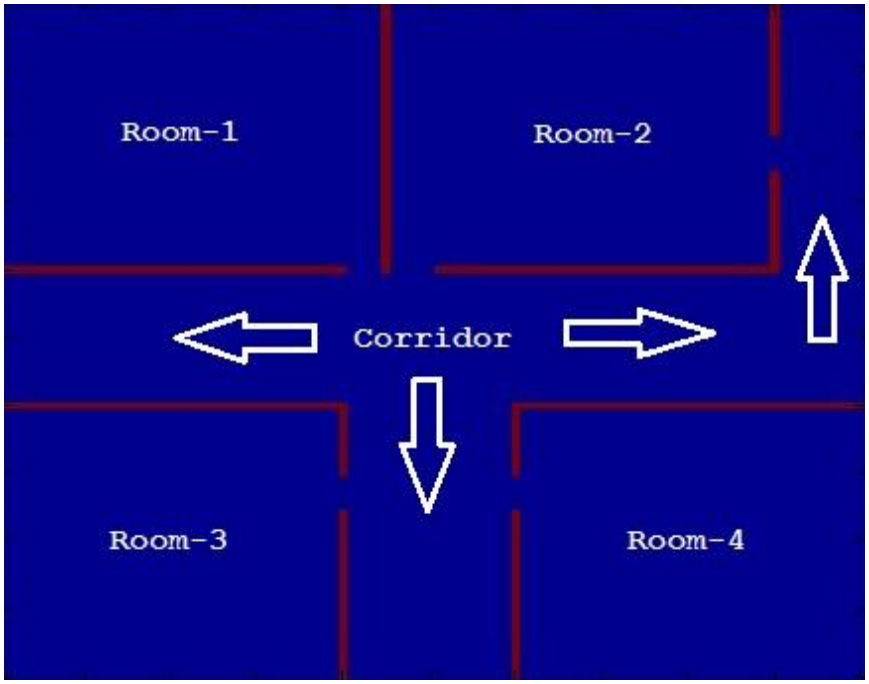


Figure 6.19 Banquet Hall with four large rooms and corridor

It can be observed from Table 6.7 that the standard deviation of the areas of different partitions in three different maps and for different size of the robot team is significantly lesser for the proposed approach when compared with the *KH*-Algorithm. The low standard deviation indicates that the proposed approach creates balanced partitions.

Tables 6.1, 6.2, and 6.3 show the area of the biggest partition (*MAX_AREA*) and the area of the smallest partition (*MIN_AREA*) for the proposed approach for three different maps and for different sizes of the robot team. In the case of *KH*-Algorithm, for hundred simulation runs and random placement of robots we have computed the average area of the biggest partition, as shown in Table 6.5 and the average area of the smallest partition as shown in Table 6.6. The primary requirement of balanced partitioning is that the difference between the area of the biggest partition and all the other partitions should be minimum. For the proposed approach and the *KH*-Algorithm the difference between the areas of the biggest partition and the smallest partition for the three different maps is computed after varying the number of robots from R = 4 to R = 7, as shown in Table 6.8. It is clearly evident that this difference is significantly lower when compared with the *KH*-Algorithm. This difference is a good indicator of imbalance in the *KH*-Algorithm and better balancing in case of the proposed approach.

Table 6.1 Standard Deviation, Area Of The Biggest Partition, And Area Of The Smallest Partition For Hospital Map

		R=4	R=5	R=6	R=7
HOSPITAL (all partition sizes are in m^2)	P1	3729	4174	3625	1403
	P2	5977	2801	3586	2627
	P3	4339	3586	1341	2936
	P4	3625	4482	3350	1341
	P5		2627	3141	2388
	P6			2627	3625
	P7				3350
	SD	1086.32	817.15	865.99	889.71
	MAX_AREA	5977	4482	3625	3625
	MIN_AREA	3625	2627	1341	1341

Table 6.2 Standard Deviation, Area Of The Biggest Partition, And Area Of The Smallest Partition For Office Map

		R=4	R=5	R=6	R=7
OFFICE (all partition sizes are in m^2)	P1	2713	2153	1848	951
	P2	2618	2075	1911	1962
	P3	1962	1103	1962	1108
	P4	1848	1848	1399	1046
	P5		1962	1314	1399
	P6			707	827
	P7				1848
	SD	443.23	421.49	484.52	446.28
	MAX_AREA	2713	2153	1962	1962
	MIN_AREA	1848	1103	707	827

Table 6.3 Standard Deviation, Area Of The Biggest Partition, And Area Of The Smallest Partition For Banquet Hall Map

		R=4	R=5	R=6	R=7
BANQUET HALL (all partition sizes are in m^2)	P1	2249	2842	2644	3178
	P2	1605	1605	1692	1223
	P3	2644	1779	1799	1799
	P4	3178	1692	1522	1560
	P5		1758	1605	1375
	P6			1320	1128
	P7				1692
	SD	662.87	511.43	172.64	313.48
	MAX_AREA	3178	2842	1799	1799
	MIN_AREA	1605	1605	1320	919

Table 6.4 Average Standard deviation of Partitions of Hundread Simulation Runs of the KH-Algorithm

Average Standard Deviation for *KH*-Algorithm				
No. of Robots	R=4	R=5	R=6	R=7
Hospital Map	2959.5	2259.4	2214.8	2040
Office Map	1312.3	1140	1154	921.58
Banquet Hall	1619	1270	1218.7	849.2

Table 6.5 Average of the Areas of the Biggest Partition for Hundread Random Placements of Robots in Three Different Maps

Average *MAX_AREA* for *KH*-Algorithm (all partition sizes are in m^2)				
No. of Robots	R=4	R=5	R=6	R=7
Hospital Map	7747.1	7072.5	6147.9	6130.5
Office Map	3690.9	3244	3237	2820.3
Banquet Hall	4394	3755	3755	2975

Table 6.6 Average of the Areas of Smallest Partition for Hundread Random Placements of Robots in Three Different Maps

Average *MIN_AREA* for *KH*-Algorithm (all partition sizes are in m^2)				
No. of Robots	R=4	R=5	R=6	R=7
Hospital Map	1432.8	1296.4	581.3	583.3
Office Map	765.8	570.4	359	293.7
Banquet Hall	813.3	781.5	514.1	474.6

Table 6.7 Comparision Based on the Standard Deviation of the Partitioned Areas

No. of Robots	**Hospital**		**Office**		**Banquet Hall**	
	KH	**PROPOSED**	**KH**	**PROPOSED**	**KH**	**PROPOSED**
R=7	2040	889.71	921.58	446.28	849.2	313.48
R=6	2214.8	865.99	1154	484.52	1218.7	172.64
R=5	2259.4	817.15	1140	421.49	1270	511.43
R=4	2959.5	1086.32	1312.3	443.23	1619	662.87

Table 6.8 Comparision of the Difference Between the Areas of the Biggest and the Smallest Partition

No. of Robots		**Hospital**		**Office**		**Banquet Hall**	
		KH	**PROPOSED**	**KH**	**PROPOSED**	**KH**	**PROPOSED**
All partition sizes are in m^2	R=7	5547.2	2352	2526.6	865	2500.4	1573
	R=6	5566.6	1855	2878.2	1050	3240.9	1237
	R=5	5776.1	2284	2673.9	1255	2973.4	459
	R=4	6314.3	2284	2925.1	1135	3580.6	860

6.9 CHAPTER SUMMARY

In this chapter we considered the task of partitioning of an unknown region into sub-regions which are then apportioned to a multi-robot system for further processing. It is evident from the literature that polygonal decomposition of workspace is superior than the grid based decomposition. Many existing approaches are based on Voronoi partitioning. However, they produce unbalanced partitions resulting in uneven distribution of the workload to individual robots. The approach proposed in this chapter creates balanced partitions of the unknown region so that the regions to be processed by individual robots are almost equal in area. This type of partitioning can be especially useful in applications like floor cleaning, surveillance, and patrolling, wherein fair distribution of the workload between the agents is important. The proposed approach requires the robots to use minimalistic sensors to determine the boundaries of the unknown region which is then converted into a weighted connected graph. This graph is then partitioned into maximally balanced sub-graphs using genetic algorithm. These sub-graphs are then optimally assigned to the available robots. The proposed approach makes it possible to create balanced partitions of the unknown environment by allowing the robot team to coordinate and exchange different forms of information (bounds, inaccessible portions of the partitions of each robot, size of the partitions etc.) about the unknown environment. This information is locally generated/ discovered by the robots. The robots communicate with their peers for creating load balanced partitions by way of auctioning. The proposed approach substantiates the argument that an efficient coordination algorithm improves the efficiency of the multi-robot system in solving a common objective.

Chapter 7

Conclusion and Future Scope of Work

7.1 CONCLUSIONS

In many situations it has been observed that multiple robots cooperate to perform complex tasks that would otherwise be impossible or difficult for one single powerful robot to accomplish. The concept behind using multi-robot systems for solving complex problems requires smaller sub-problems to be assigned to individual robots while allowing the robots to interact with each other for sharing information. Simple robots can be built easily and made to cooperate with each other to achieve some common objective. Multi-robot systems are very cost effective compared to building a single costly robot with many capabilities. As these multi-robot systems are usually decentralized and inherently redundant, they are fault tolerant and thus improve the reliability and robustness of the system. The simplicity of multi-robot systems has led to its wide set of applications. However, this simplicity also brings additional challenges in setting up and deploying such systems. In this thesis, we have studied coordination algorithms for multi-robot systems in three different domains namely, g*eometric pattern formation, online terrain coverage,* and *balanced area partitioning/ decomposition.* The primary contributions of the thesis for each of these domain are listed below.

Geometric pattern formation by multiple mobile robots is a problem that has gained considerable attention because of its wide applicability in variety of tasks i.e. multi-robot formations can act as sensor arrays for monitoring the state of the environment, area exploration etc. In fact, specific roles can be assigned to individual robots if they can in a decentralized manner develop common consensus on the position of individual robots and their peers. In this respect, uniform circle formation problem has been investigated by the researchers in theoretical computer science. Simplified assumptions have been considered for designing distributed solutions that are proven to be sound and complete. These algorithms have not been validated experimentally and therefore cannot be compared with other empirical approaches. Following are the contributions of the thesis in solving the geometric pattern formation problem:

(a) Found approximate solutions to various abstract assumptions considered in one of the representative algorithms i.e. the Défago and Konagaya's (DK) algorithm [Défago 2002, Défago 2008] for solving the uniform circle formation problem.

(b) The DK algorithm is experimentally validated on a multi-robot test-bed comprising of five e-puck robots.

(c) Proposed a distributed algorithm i.e. the STATE algorithm [Gautam 2016a] for solving the uniform circle formation problem. The STATE algorithm has also been implemented on our experimental test-bed and was found to perform better than the DK algorithm in terms of reduced number of activation steps and reduced time of convergence. Two important findings from the experimental results are as follows:

- ✓ Irrespective of the activation probability and the placement scenario chosen for the DK algorithm, the percentage improvement achieved by the STATE algorithm over DK algorithm increases as the size of the circle increases. For instance, for a activation probability P_r = 0.5 and worst placement scenario for DK algorithm, the percentage improvement achieved by the STATE algorithm (in terms of average activation steps) is 27.6%, 29.14%, and 31.18% and it is 18.32%, 22.75%, and 27.42% in terms of average completion time for the circles with diameter of 1 meter, 2 meters, and 3 meters respectively.
- ✓ Irrespective of the activation probability chosen for the DK algorithm and the size of the circle to be formed, the percentage improvement shown by the STATE algorithm over DK algorithm increases when the initial placement of the robots is made worse i.e. from best to random and from random to worse placement. For example, if we consider a large circle with a diameter of 3 meters and fix the activation probability P_r = 0.5 for the DK algorithm, the STATE algorithm performs 25.18%, 28.97%, and 31.18% better (in terms of average activation steps) for the best, random and worst placement respectively. Further the STATE algorithm performs 20.66%, 22.59%, and 27.42% better (in terms of average completion time) than the DK algorithm for the best, random and worst placement.

(d) Setup a multi-robot test-bed that provides real-time localization of mobile robots.

(e) Proposed a scheme for asynchronous and non-blocking inter-robot communication.

Online terrain coverage is yet another critical task which can be solved by using multi-robot systems. The primary requirement of the task is to completely cover an unknown terrain. Frontier based exploration has become a de-facto standard in multi-robot exploration and coverage. Under this head

(a) We have identified the limitations of various frontier propagation algorithms for online terrain coverage.

(b) We have suggested two online terrain coverage algorithms. The first algorithm is a centralized algorithm [Gautam 2015] referred to as Cluster, Allocate, Cover (CAC) and the second algorithm is completely decentralized algorithm [Gautam 2016b] referred to as Frontier Allocation Synchronized by Token Passing (FAST). The CAC algorithm is shown to perform better than some of the state of the art algorithms [Yamauchi 1998, Burgard 2005, Puig 2011] both in simulation and on a multi-robot test-bed. For example, in an outdoor map using a team of six robots the percentage improvement shown by CAC (in terms of completion time) over [Burgard 2005], [Puig 2011], and [Yamauchi 1998] is 25.94%, 18.82%, and 13.97% respectively. Also, the percentage improvement of CAC (in terms of non-overlapping coverage) over [Burgard 2005], [Puig 2011], and [Yamauchi 1998] is 54.28%, 44%, and 35.51% respectively. The dispersion of mobile robots is an emergent phenomenon in CAC and therefore it is also suitable for multi-robot exploration task. One limitation of CAC is that it is a centralized approach and is not robust to failure of robots. FAST overcomes this limitation of CAC as it deals with robot failures in a graceful manner. The algorithm FAST is based on the concept of following structured trajectories that is proven to be a powerful approach for terrain coverage in the literature.

(c) FAST and other representative approaches like [Gerlein 2011], referred to as BSA-CM, and its extension that is proposed in this thesis, referred to as BSA-CMI, are implemented both in simulation and on a multi-robot test-bed. FAST is shown to outperform all the other approaches both in terms of reduced time to complete coverage and reduced overlapping coverage whereas BSA-CMI works better than original BSA-CM and the CAC algorithm. For instance, in a Living Room map using a team of five robots the percentage improvement shown by FAST (in terms of completion time) over CAC, BSA-CM, and BSA-CMI is 23.8%, 16.9%, and 8.43% respectively. Also, the percentage improvement of FAST (in terms of non-overlapping coverage) over CAC, BSA-CM, and BSA-CMI is 20%, 14.28%, and 7.69% respectively.

Area partitioning/ decomposition is one of the fundamental tasks in mobile robotics. It is important to build and maintain the map of the environment for a mobile robot to successfully navigate and plan its path from source to destination. The map thus created is represented in the robot's memory in either of the two forms (a) grid based maps or (b) topological maps. It is easier to obtain, represent, and maintain grid based maps but they have a limitation in the sense that most planning and navigation algorithms using grid based maps have high memory and computational requirements. On the other hand, if polygonal decomposition of the environment is done it can easily be translated into topological maps that are graph like maps allowing more efficient planning and navigation. Many recent state-of-the-art approaches have employed polygonal decomposition of the environment based on Voronoi diagrams [Wu 2007, Wurm 2008, Hungerford 2014] and Delaunay triangulation [Fazli 2010, Fazli 2013] and constructed a topological map of the environment. Later, the map is segmented into different partitions that are allocated to different robots for processing. In this thesis

(a) The problem of balanced partitioning of the unknown region using multi-robot system is addressed. The environment decomposition method and the task allocation method that are used by previous approaches results in unbalanced partitions.

(b) We have proposed an algorithm that determines the skeleton of the unknown region first and then creates a topological representation of the environment. This topological graph is partitioned using genetic algorithm into as many maximally balanced partitions as the number of robots. These partitions can be optimally allocated to the individual robots.

(c) The proposed approach for balanced partitioning is compared in simulation with one of the most recent approaches [Hungerford 2016], referred to as *KH* algorithm. The *KH* algorithm uses Voronoi diagrams for partitioning the environment for a multi-robot coverage task. The proposed approach is found to perform better than the *KH* algorithm in terms of creating balanced partitions and therefore ensuring better utilization of the multi-robot system. Smaller the standard deviation of the areas of all the partitions, better the partitions are balanced. For instance, in a hospital map for seven robots, seven partitions are created. The standard deviation of the partitions created by the proposed algorithm is 889.71 and that of the KH algorithm is 2040 which is a significant difference. Simulation results in maps of different sizes and robot team size substantiate our claim of balanced partitioning.

This thesis extends the state-of-the-art in three different multi-robot application domains. The algorithms proposed in this thesis have been compared with some of the representative state-of-the-art approaches and are shown to perform better on different cost measures both in simulation and on a multi-robot testbed. It is shown that efficient coordination amongst the robots helps in finishing the task in lesser time while reducing resource wastage/ underutilization.

7.2 SCOPE FOR FUTURE WORK

The problem of Geometric pattern formation requires the robots to accurately localize their peers. Multi-robot localization for solving the arbitrary pattern formation problem demands attention. Neighbor referenced [Fredslund 2002, Mead 2009] and leader-follower [Monteiro 2010] approaches have been successfully tested in the past. The efficiency of these algorithms in terms of energy consumptions is required to be quantified.

Limited communication range of robots and its impact on the performance of the coverage algorithm is one aspect we have not explored in this thesis. Although we have proposed a fault tolerant algorithm (FAST) for online coverage of a terrain but it is designed for the purpose of indoor office like maps and tested in a controlled laboratory setting. The approach can be extended to account for message losses due to noise in the environment.

The proposed algorithm for load balanced partitioning of the workspace needs to be implemented on a team of physical mobile robots. Multiple robots can be dispatched for the purpose of exploration of unknown region. The effect of balanced partitioning on the performance of the exploration algorithm can be quantitatively measured and compared with other approaches. Also fault-tolerance is a vital aspect that is required to be incorporated in the load balancing algorithm.

REFERENCES

1. [Agmon 2008] Agmon, N., Hazon, N., & Kaminka, G. (2008). The giving tree: constructing trees for efficient offline and online multi-robot coverage. Annals of Mathematics and Artificial Intelligence, 52(2-4), 143-168. http://dx.doi.org/10.1007/s10472-009-9121-1

2. [Amato 2015] Amato, C., Konidaris, G., Anders, A., Cruz, G., How, J., & Kaelbling, L. (2015). Policy Search for Multi-Robot Coordination under Uncertainty. Robotics: Science and Systems XI. http://dx.doi.org/10.15607/rss.2015.xi.007

3. [Ando 1999] Ando, H., Oasa, Y., Suzuki, I., & Yamashita, M. (1999). Distributed memoryless point convergence algorithm for mobile robots with limited visibility. IEEE Trans. Robot. Automat., 15(5), 818-828. http://dx.doi.org/10.1109/70.795787

4. [Andries 2013] Andries, M. & Charpillet, F. (2013). Multi-robot exploration of unknown environments with identification of exploration completion and post-exploration rendezvous using ant algorithms. 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. http://dx.doi.org/10.1109/iros.2013.6697164

5. [Antonelli 2007] Antonelli, G., Arrichiello, F., Chakraborti, S., & Chiaverini, S. (2007). Experiences of formation control of multi-robot systems with the Null-Space-based Behavioral Control. 2007 IEEE International Conference on Robotics and Automation. http://dx.doi.org/10.1109/robot.2007.363126

6. [Antonelli 2009a] Antonelli, G., Arrichiello, F., & Chiaverini, S. (2009). Experiments of Formation Control with Multirobot Systems Using the Null-Space-Based Behavioral Control. IEEE Transactions on Control Systems Technology, 17(5), 1173-1182. http://dx.doi.org/10.1109/tcst.2008.2004447

7. [Antonelli 2009b] Antonelli, G., Arrichiello, F., & Chiaverini, S. (2009). Flocking for multi-robot systems via the Null-Space-based Behavioral control. Swarm Intell, 4(1), 37-56. http://dx.doi.org/10.1007/s11721-009-0036-6

8. [Arkin 1998] Arkin, R. (1998). Behavior-based robotics. Cambridge, Mass.: MIT Press.

9. [Asgharbeygi 2008] Asgharbeygi, N. & Maleki, A. (2008). Geodesic K-means clustering. 2008 19th International Conference on Pattern Recognition. http://dx.doi.org/10.1109/icpr.2008.4761241

10. [Durrant-Whyte 2006] Durrant-Whyte, H. & Bailey, T. (2006). Simultaneous localization and mapping: part I. IEEE Robotics & Automation Magazine, 13(2), 99-110. http://dx.doi.org/10.1109/mra.2006.1638022.

11. [Bailey 2006] Bailey, T. & Durrant-Whyte, H. (2006). Simultaneous localization and mapping (SLAM): part II. IEEE Robotics & Automation Magazine, 13(3), 108-117. http://dx.doi.org/10.1109/mra.2006.1678144

12. [Balch 1998] Balch, T. & Arkin, R. (1998). Behavior-based formation control for multirobot teams. IEEE Trans. Robot. Automat., 14(6), 926-939. http://dx.doi.org/10.1109/70.736776

13. [Balch 1999] Balch, T. (1999). The impact of diversity on performance in multirobot foraging. Autonomous Agents.

14. [Barnes 1991] Barnes, D. & Gray, J. (1991). Behavior synthesis for cooperative mobile robot control. International Conference on Control.

15. [Boada 1998] Boada, B., Moreno, L., & Salichs, M. (1998). Sensor Coordination for Multi Mobile Robots Systems. Distributed Autonomous Robotic Systems 3, 13-22. http://dx.doi.org/10.1007/978-3-642-72198-4_2

16. [Bormann 2016] Bormann, R., Jordan, F., Li, W., Hampp, J., & Hagele, M. (2016). Room segmentation: Survey, implementation, and analysis. 2016 IEEE International Conference on Robotics and Automation (ICRA). http://dx.doi.org/10.1109/icra.2016.7487234

17. [Breitenmoser 2010] Breitenmoser, A., Schwager, M., Metzger, J., Siegwart, R., & Rus, D. (2010). Voronoi coverage of non-convex environments with a group of networked robots. 2010 IEEE International Conference on Robotics and Automation. http://dx.doi.org/10.1109/robot.2010.5509696

18. [Bruce 2000] Bruce, J., Balch, T., & Veloso, M. (2000) Fast and inexpensive color image segmentation for interactive robots. Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113). http://dx.doi.org/10.1109/iros.2000.895274

19. [Burgard 2005] Burgard, W., Moors, M., Stachniss, C., & Schneider, F. (2005). Coordinated multi-robot exploration. IEEE Trans. Robot., 21(3), 376-386. http://dx.doi.org/10.1109/tro.2004.839232

20. [Candea 2001] Candea, C., Hu, H., Iocchi, L., Nardi, D., & Piaggio, M. (2001). Coordination in multi-agent RoboCup teams. Robotics and Autonomous Systems, 36(2-3), 67-86. http://dx.doi.org/10.1016/s0921-8890(01)00137-3

21. [Cannata 2011] Cannata, G. & Sgorbissa, A. (2011). A Minimalist Algorithm for Multirobot Continuous Coverage. IEEE Trans. Robot., 27(2), 297-312. http://dx.doi.org/10.1109/tro.2011.2104510

22. [Cao 1997] Cao, Y., Fukunaga, A., & Kahng, A. (1997). Autonomous Robots, 4(1), 7-27. http://dx.doi.org/10.1023/a:1008855018923

23. [Carvalho 2013] Carvalho, F., Cavalcante, R., Vieira, M., Chaimowicz, L., & Campos, M. (2013). A Multi-robot Exploration Approach Based on Distributed Graph Coloring. 2013 Latin American Robotics Symposium and Competition. http://dx.doi.org/10.1109/lars.2013.71

24. [Castello 2013] Castello, E., Yamamoto, T., Nakamura, Y., & Ishiguro, H. (2013). Task Allocation for a robotic swarm based on an Adaptive Response Threshold Model. 2013 13th International Conference on Control, Automation and Systems (ICCAS 2013). http://dx.doi.org/10.1109/iccas.2013.6703905

25. [Chand 2013] Chand, P. & Carnegie, D. (2013). Mapping and exploration in a hierarchical heterogeneous multi-robot system using limited capability robots. Robotics and Autonomous Systems, 61(6), 565-579. http://dx.doi.org/10.1016/j.robot.2013.02.009

26. [Chang 2004] Chang, F., Chen, C., & Lu, C. (2004). A linear-time component-labeling algorithm using contour tracing technique. Computer Vision and Image Understanding, 93(2), 206-220. http://dx.doi.org/10.1016/j.cviu.2003.09.002

27. [Chatzigiannakis 2004] Chatzigiannakis, I., Markou, M., & Nikoletseas, S. (2004). Distributed Circle Formation for Anonymous Oblivious Robots. Experimental and Efficient Algorithms, 159-174. http://dx.doi.org/10.1007/978-3-540-24838-5_12

28. [Cheikhrouhoua 2014] Cheikhrouhou, O., Koubaa, A., & Bennaceur, H. (2014). Move and improve: A distributed multi-robot coordination approach for multiple

depots multiple travelling salesmen problem. 2014 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC). http://dx.doi.org/10.1109/icarsc.2014.6849758

29. [Chlebíková 1996] Chlebíková, J. (1996). Approximating the maximally balanced connected partition problem in graphs. Information Processing Letters, 60(5), 225-230. http://dx.doi.org/10.1016/s0020-0190(96)00175-5

30. [Choset 2005] Choset, H. (2005). Principles of robot motion. Cambridge, Mass.: MIT Press.

31. [Christensen 2009] Christensen, A., O'Grady, R., & Dorigo, M. (2009). From Fireflies to Fault-Tolerant Swarms of Robots. IEEE Transactions on Evolutionary Computation, 13(4), 754-766. http://dx.doi.org/10.1109/tevc.2009.2017516

32. [Cianci 2006] Cianci CM, Raemy X, Pugh J, Martinoli A (2006) Communication in a swarm of miniature robots: the e-puck as an educational tool for swarm robotics. Infoscience, EPFL's scientific publications. https://infoscience.epfl.ch/record/100015/files/44330103.pdf. Accessed 13 June 2016

33. [Cincotti 2002] Cincotti, A., Cutello, V., & Pavone, M. (2002). Graph partitioning using genetic algorithms with ODPX. 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600). http://dx.doi.org/10.1109/cec.2002.1006268

34. [Damer 2006] Damer, S., Ludwig, L., LaPoint, M., Gini, M., Papanikolopoulos, N., & Budenske, J. (2006). Dispersion and exploration algorithms for robots in unknown environments. Unmanned Systems Technology VIII. http://dx.doi.org/10.1117/12.668915

35. [Dasgupta 2009] Dasgupta, P. and Cheng, K. (2009). Distributed coverage of unknown environments using multi-robot swarms with memory and communication constraints. Technical Report No. cst-2009-1, Department of Computer Science, University of Nebraska at Omaha.

36. [Debest 1995] Debest, XA. (1995). Remark about self-stabilizing systems. Communications of the ACM, 38(2), 115-117.

37. [Défago 2002] Défago, X. & Konagaya, A. (2002). Circle formation for oblivious anonymous mobile robots with no common sense of orientation. Second ACM

International Workshop On Principles of Mobile Computing - POMC '02. http://dx.doi.org/10.1145/584490.584509

38. [Défago 2008] Défago, X. & Souissi, S. (2008). Non-uniform circle formation algorithm for oblivious mobile robots with convergence toward uniformity. Theoretical Computer Science, 396(1-3), 97-112. http://dx.doi.org/10.1016/j.tcs.2008.01.050

39. [Dieudonné 2007] Dieudonné, Y. & Petit, F. (2007). Circle formation of weak robots and Lyndon words. Information Processing Letters, 101(4), 156-162. http://dx.doi.org/10.1016/j.ipl.2006.09.008

40. [Dieudonné 2008] Dieudonné, Y., Labbani-Igbida, O., & Petit, F. (2008). Circle formation of weak mobile robots. ACM Trans. Auton. Adapt. Syst., 3(4), 1-20. http://dx.doi.org/10.1145/1452001.1452006

41. [Dieudonné 2009] Dieudonné, Y., Dolev, S., Petit, F., & Segal, M. (2009). Brief announcement. 28th ACM Symposium on Principles of Distributed Computing - PODC '09. http://dx.doi.org/10.1145/1582716.1582781

42. [Dijkstra 1959] Dijkstra, E. (1959). A note on two problems in connexion with graphs. Numer. Math., 1(1), 269-271. http://dx.doi.org/10.1007/bf01386390

43. [Dissanayake 2001] Dissanayake, M., Newman, P., Clark, S., Durrant-Whyte, H., & Csorba, M. (2001). A solution to the simultaneous localization and map building (SLAM) problem. IEEE Trans. Robot. Automat., 17(3), 229-241. http://dx.doi.org/10.1109/70.938381

44. [Ducatelle 2009] Ducatelle, F., Forster, A., Caro, G. D., and Gambardella, L. (2009). New task allocation methods for robotic swarms. 9th IEEE/RAS Conference on Autonomous Robot Systems and Competitions

45. [Dudek 1996] Dudek, G., Jenkin, M., Milios, E., & Wilkes, D. (1996). A taxonomy for multi-agent robotics. Autonomous Robots, 3(4). http://dx.doi.org/10.1007/bf00240651

46. [Elfes 1989] Elfes, A. (1989). Using occupancy grids for mobile robot perception and navigation. Computer, 22(6), 46-57. http://dx.doi.org/10.1109/2.30720

47. [Elor 2011] Elor, Y. & Bruckstein, A. (2011). Uniform multi-agent deployment on a ring. Theoretical Computer Science, 412(8-10), 783-795. http://dx.doi.org/10.1016/j.tcs.2010.11.023

48. [Fabrice 1993] Noreils, F. (1993). Toward a Robot Architecture Integrating Cooperation between Mobile Robots: Application to Indoor Environment. The International Journal of Robotics Research, 12(1), 79-98. http://dx.doi.org/10.1177/027836499301200106

49. [Fagiolini 2008] Fagiolini, A., Pellinacci, M., Valenti, G., Dini, G., & Bicchi, A. (2008). Consensus-based distributed intrusion detection for multi-robot systems. 2008 IEEE International Conference on Robotics and Automation. http://dx.doi.org/10.1109/robot.2008.4543196

50. [Fagiolini 2009] Fagiolini, A., Babboni, F., & Bicchi, A. (2009). Dynamic distributed intrusion detection for secure multi-robot systems. 2009 IEEE International Conference on Robotics and Automation. http://dx.doi.org/10.1109/robot.2009.5152608

51. [Farinelli 2004] Farinelli, A., Iocchi, L., & Nardi, D. (2004). Multirobot Systems: A Classification Focused on Coordination. IEEE Trans. Syst., Man, Cybern. B, 34(5), 2015-2028. http://dx.doi.org/10.1109/tsmcb.2004.832155

52. [Fax 2002] Fax, J. & Murray, R. (2004). Information Flow and Cooperative Control of Vehicle Formations. IEEE Transactions On Automatic Control, 49(9), 1465-1476. http://dx.doi.org/10.1109/tac.2004.834433

53. [Fazli 2010] Fazli P, Davoodi A, Pasquier P, Mackworth AK. (2010). Fault-Tolerant Multi-Robot Area Coverage with Limited Visibility. In: ICRA Workshop: Search and Pursuit/Evasion in the Physical World

54. [Fazli 2013] Fazli, P., Davoodi, A., & Mackworth, A. (2013). Multi-robot repeated area coverage. Autonomous Robots, 34(4), 251-276. http://dx.doi.org/10.1007/s10514-012-9319-7

55. [Ferranti 2007] Ferranti, E., Trigoni, N., & Levene, M. (2007). Brick& Mortar: an on-line multi-agent exploration algorithm. 2007 IEEE International Conference on Robotics and Automation. http://dx.doi.org/10.1109/robot.2007.363078

56. [Ferranti 2009] Ferranti, E., Trigoni, N., & Levene, M. (2008). Rapid exploration of unknown areas through dynamic deployment of mobile and stationary sensor nodes. Auton Agent Multi-Agent Syst, 19(2), 210-243. http://dx.doi.org/10.1007/s10458-008-9075-4

57. [Firebird-V 2016] Fire Bird V ATMEGA2560 Robotic Research Platform - Nex Robotics. (2016). Nex-robotics.com. Retrieved 21 August 2016, from http://www.nex-robotics.com/products/fire-bird-v-robots/fire-bird-v-atmega2560-robotic-research-platform.html

58. [Flocchini 1999] Flocchini, P., Prencipe, G., Santoro, N., & Widmayer, P. (1999). Hard Tasks for Weak Robots: The Role of Common Knowledge in Pattern Formation by Autonomous Mobile Robots. Algorithms and Computation, 93-102. http://dx.doi.org/10.1007/3-540-46632-0_10

59. [Flocchini 2005] Flocchini, P., Prencipe, G., Santoro, N., & Widmayer, P. (2005). Gathering of asynchronous robots with limited visibility. Theoretical Computer Science, 337(1-3), 147-168. http://dx.doi.org/10.1016/j.tcs.2005.01.001

60. [Flocchini 2008] Flocchini, P., Prencipe, G., & Santoro, N. (2008). Self-deployment of mobile sensors on a ring. Theoretical Computer Science, 402(1), 67-80. http://dx.doi.org/10.1016/j.tcs.2008.03.006

61. [Fredslund 2002] Fredslund, J. & Mataric, M. (2002). A general algorithm for robot formations using local sensing and minimal communication. IEEE Trans. Robot. Automat., 18(5), 837-846. http://dx.doi.org/10.1109/tra.2002.803458

62. [Fu 2009] Fu, J., Bandyopadhyay, T., & Ang, M. (2009). Local Voronoi Decomposition for multi-agent task allocation. 2009 IEEE International Conference on Robotics and Automation. http://dx.doi.org/10.1109/robot.2009.5152829

63. [Galceran 2013] Galceran, E. & Carreras, M. (2013). A survey on coverage path planning for robotics. Robotics and Autonomous Systems, 61(12), 1258-1276. http://dx.doi.org/10.1016/j.robot.2013.09.004

64. [Gamma 1995] Gamma, E. (1995). Design patterns. Reading, Mass.: Addison-Wesley.

65. [Gao 2006] Gao, P. & Cai, Z. (2006). Multi-robot task allocation for exploration. Journal of Central South University of Technology, 13(5), 548-551. http://dx.doi.org/10.1007/s11771-006-0085-6

66. [Garg 2004] Garg VK (2004) Resource Allocation. In: Concurrent and Distributed Computing in Java, Wiley-IEEE Press, pp 138-141.

67. [Gautam 2012] Gautam, A., Mohan, S., & Misra, J. (2012). A practical framework for uniform circle formation by multiple mobile robots. 7th IEEE International Conference on Industrial and Information Systems. http://dx.doi.org/10.1109/iciinfs.2012.6304765

68. [Gautam 2013a] Gautam, A., Mohan, S., & Shekhawat, V. (2013). A token passing approach for circle formation by multiple mobile robots. 6th IEEE International Conference on Contemporary Computing. http://dx.doi.org/10.1109/ic3.2013.6612251

69. [Gautam 2013b] Gautam, A. & Mohan, S. (2013). A distributed algorithm for circle formation by multiple mobile robots. IEEE International Conference On Control, Automation, Robotics and Embedded Systems. http://dx.doi.org/10.1109/care.2013.6733699

70. [Gautam 2014] Gautam, A., Saxena, A., Mall, P., & Mohan, S. (2014). Positioning multiple mobile robots for geometric pattern formation: An empirical analysis. Seventh IEEE International Conference on Contemporary Computing. http://dx.doi.org/10.1109/ic3.2014.6897242

71. [Gautam 2015] Gautam, A., Murthy, J., Kumar, G., Ram, S., Jha, B., & Mohan, S. (2015). Cluster, Allocate, Cover: An Efficient Approach for Multi-Robot Coverage. IEEE International Conference on Systems, Man, and Cybernetics. http://dx.doi.org/10.1109/smc.2015.47

72. [Gautam 2016a] Gautam, A. & Mohan, S. (2016). STATE: Distributed Algorithm for Uniform Circle Formation by Multiple Mobile Robots. Journal of Intelligent Service Robotics. http://dx.doi.org/10.1007/s11370-016-0205-6

73. [Gautam 2016b] Gautam, A., Jha, B., Kumar, G., Murthy, J. K., Ram, S. P. A. & Mohan, S. (2016). FAST: Synchronous Frontier Allocation for Scalable Online

Multi-Robot Terrain Coverage. Journal of Intelligent Robotic Systems. http://dx.doi.org/10.1007/s10846-016-0416-2

74. [Gautam 2016c] Gautam, A., Thakur, A., Dhanania, G., & Mohan, S. (2016). A Distributed Algorithm for Balanced Multi-Robot Task Allocation. In *2016 IEEE 11th International Conference on Industrial and Information Systems (ICIIS 2016).*

75. [Gerkey 2002] Gerkey, B. & Mataric, M. (2002). Sold!: auction methods for multirobot coordination. IEEE Trans. Robot. Automat., 18(5), 758-768. http://dx.doi.org/10.1109/tra.2002.803462

76. [Gerkey 2004a] Gerkey, B. (2004). A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems. The International Journal of Robotics Research, 23(9), 939-954. http://dx.doi.org/10.1177/0278364904045564

77. [Gerkey 2004d] Gerkey, B.P. and Mataric, M.J. (2004). A formal framework for the study of task allocation in multi-robot systems. International Journal of Robotics Research, vol. 23(9), pp. 939–954, 2004.

78. [Gerlein 2011] Gerlein, E. & Gonzalez, E. (2011). Multirobot Cooperative Model Applied to Coverage of Unknown Regions. Multi-Robot Systems, Trends and Development. http://dx.doi.org/10.5772/13241

79. [Giordani 2010] Giordani, S., Lujak, M., & Martinelli, F. (2010). A Distributed Algorithm for the Multi-Robot Task Allocation Problem. Trends in Applied Intelligent Systems, 721-730. http://dx.doi.org/10.1007/978-3-642-13022-9_72

80. [Goldberg 1997] Goldberg, D. and Mataric, M. (1997). Interference as a tool for designing and evaluating multi-robot controllers. In Proceedings, AAAI-97

81. [Gonzalez 2003] Gonzalez, E., Alarcon, M., Aristizabal, P., & Parra, C. BSA: a coverage algorithm. 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453). http://dx.doi.org/10.1109/iros.2003.1248885

82. [GraphStream 2016] GraphStream - GraphStream - A Dynamic Graph Library. (2016). Graphstream-project.org. Retrieved 21 August 2016, from http://graphstream-project.org/

83. [Gutiérrez 2009] Gutierrez, A., Campo, A., Dorigo, M., Donate, J., Monasterio-Huelin, F., & Magdalena, L. (2009). Open E-puck Range & Bearing

miniaturized board for local communication in swarm robotics. 2009 IEEE International Conference on Robotics and Automation. http://dx.doi.org/10.1109/robot.2009.5152456

84. [Guzzoni 1997] Guzzoni, D., Cheyer, A., Julia, L., & Konolige, K. (1997). Many Robots Make Short Work: Report of the SRI International Mobile Robot Team. AI Magazine, 18(1), 55. Retrieved from http://dx.doi.org/10.1609/aimag.v18i1.1274

85. [Hao 2005] Hao, Y. & Agrawal, S. (2005). Planning and control of UGV formations in a dynamic environment: A practical framework with experiments. Robotics and Autonomous Systems, 51(2-3), 101-110. http://dx.doi.org/10.1016/j.robot.2005.01.001

86. [Harabor 2011] Harabor, D.D., Grastien, A. (2011). Online graph pruning for path nding on grid maps. In Proceedings, AAAI-2011

87. [Haumann 2010] Haumann, A., Listmann, K., & Willert, V. (2010). DisCoverage: A new paradigm for multi-robot exploration. 2010 IEEE International Conference on Robotics and Automation. http://dx.doi.org/10.1109/robot.2010.5509993

88. [Hazon 2005] Hazon, N. & Kaminka, G. (2005). Redundancy, Efficiency and Robustness in Multi-Robot Coverage. 2005 IEEE International Conference on Robotics and Automation. http://dx.doi.org/10.1109/robot.2005.1570205

89. [Heap 2012] Heap, B., Pagnucco, M. (2012). Repeated sequential auctions with dynamic task clusters. In Proceedings, AAAI (2012)

90. [Heap 2013] Heap, B. & Pagnucco, M. (2013). Repeated Auctions for Reallocation of Tasks with Pickup and Delivery upon Robot Failure. Lecture Notes in Computer Science, 461-469. http://dx.doi.org/10.1007/978-3-642-44927-7_35

91. [Heron USV 2016] Heron Unmanned Surface Vessel for Bathymetry. (2016). Clearpath Robotics. Retrieved 22 August 2016, from https://www.clearpathrobotics.com/heron-bathymetry-unmanned-surface-vessel/

92. [Howard 2002] Howard, A., Matarić, M., & Sukhatme, G. (2002). Mobile Sensor Network Deployment using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem. Distributed Autonomous Robotic Systems 5, 299-308. http://dx.doi.org/10.1007/978-4-431-65941-9_30

93. [Howard 2004] Howard, A. (2004). Multi-robot mapping using manifold representations. IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004. http://dx.doi.org/10.1109/robot.2004.1308933

94. [Howard 2006] Howard, A. (2006). Multi-robot Simultaneous Localization and Mapping using Particle Filters. The International Journal of Robotics Research, 25(12), 1243-1256. http://dx.doi.org/10.1177/0278364906072250

95. [Hungerford 2014] K. Hungerford, P. Dasgupta, & K. R. Guruprasad. (2014). Distributed, complete, multi-robot coverage of initially unknown environments using repartitioning. In AAMAS, IFAAMAS-2014.

96. [Hungerford 2016] Hungerford, K., Dasgupta, P., & Guruprasad, K. (2016). A Repartitioning Algorithm to Guarantee Complete, Non-overlapping Planar Coverage with Multiple Robots. Springer Tracts in Advanced Robotics, 33-48. http://dx.doi.org/10.1007/978-4-431-55879-8_3

97. [Husky UGV 2016] Husky UGV - Outdoor Field Research Robot by Clearpath. (2016). Clearpath Robotics. Retrieved 22 August 2016, from https://www.clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/

98. [Hussein 2014] Hussein, A., Adel, M., Bakr, M., Shehata, O., & Khamis, A. (2014). Multi-robot Task Allocation for Search and Rescue Missions. Journal of Physics: Conference Series, 570(5), 052006. Retrieved from http://dx.doi.org/10.1088/1742-6596/570/5/052006

99. [Juliá 2012] Juliá, M., Gil, A., & Reinoso, O. (2012). A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. Autonomous Robots, 33(4), 427-444. http://dx.doi.org/10.1007/s10514-012-9298-8

100. [Kalra 2006] Kalra, N. & Martinoli, A. (2006). Comparative Study of Market-Based and Threshold-Based Task Allocation. Distributed Autonomous Robotic Systems 7, 91-101. http://dx.doi.org/10.1007/4-431-35881-1_10

101. [Kapanoglu 2012] Kapanoglu, M., Alikalfa, M., Ozkan, M., Yazıcı, A., & Parlaktuna, O. (2010). A pattern-based genetic algorithm for multi-robot coverage path planning minimizing completion time. J Intell Manuf, 23(4), 1035-1045. http://dx.doi.org/10.1007/s10845-010-0404-5

102. [Ko 2003] Ko, J., Stewart, B., Fox, D., Konolige, K., & Limketkai, B. (2003) A practical, decision-theoretic approach to multi-robot mapping and exploration. Proceedings 2003 IEEE/RSJ International Conference On Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453). http://dx.doi.org/10.1109/iros.2003.1249654

103. [Krick 2009] Krick, L., Broucke, M., & Francis, B. (2009). Stabilisation of infinitesimally rigid formations of multi-robot networks. International Journal of Control, 82(3), 423-439. http://dx.doi.org/10.1080/00207170802108441

104. [Krishnanand 2005] Krishnanand, K. & Ghose, D. (2005). Formations of minimalist mobile robots using local-templates and spatially distributed interactions. Robotics and Autonomous Systems, 53(3-4), 194-213. http://dx.doi.org/10.1016/j.robot.2005.09.006

105. [Krontiris 2011] Krontiris, A., Louis, S., & Bekris, K. (2011). General dynamic formations for non-holonomic systems along planar curvilinear coordinates. 2011 IEEE International Conference on Robotics and Automation. http://dx.doi.org/10.1109/icra.2011.5980450

106. [Kube 1993] Kube, C. & Hong Zhang. (1993). Collective Robotics: From Social Insects to Robots. Adaptive Behavior, 2(2), 189-218. http://dx.doi.org/10.1177/105971239300200204

107. [Kuhn 1955] Kuhn, H. (1955). The Hungarian method for the assignment problem. Naval Research Logistics Quarterly, 2(1-2), 83-97. http://dx.doi.org/10.1002/nav.3800020109

108. [Latombe 1991] Latombe, J. (1991). Approximate Cell Decomposition. Robot Motion Planning, 248-294. http://dx.doi.org/10.1007/978-1-4615-4022-9_6

109. [Leonard 2010] Leonard, N., Paley, D., Davis, R., Fratantoni, D., Lekien, F., & Zhang, F. (2010). Coordinated control of an underwater glider fleet in an adaptive ocean sampling field experiment in Monterey Bay. Journal of Field Robotics, 27(6), 718-740. http://dx.doi.org/10.1002/rob.20366

110. [Liu 2014] Liu, C. (2014). Multi-Robot Task Allocation for Inspection Problems with Cooperative Tasks Using Hybrid Genetic Algorithms. Kassel, Hess: Kassel University Press.

111. [Lizi 2016] Lizi. (2016). Retrieved 30 August 2016, from http://www.gaitech.hk/proimg/RoboTiCan%20Mobile%20Platforms/Lizi%20Robot.pdf

112. [Marino 2013] Marino, A., Parker, L., Antonelli, G., & Caccavale, F. (2012). A Decentralized Architecture for Multi-Robot Systems Based on the Null-Space-Behavioral Control with Application to Multi-Robot Border Patrolling. Journal of Intelligent & Robotic Systems, 71(3-4), 423-444. http://dx.doi.org/10.1007/s10846-012-9783-5

113. [Marshall 2006] Marshall, J., Fung, T., Broucke, M., D'Eleuterio, G., & Francis, B. (2006). Experiments in multirobot coordination. Robotics and Autonomous Systems, 54(3), 265-275. http://dx.doi.org/10.1016/j.robot.2005.10.004

114. [Mataric 1995a] Matarić, M. (1995). From Local Interactions to Collective Intelligence. The Biology and Technology of Intelligent Autonomous Agents, 275-295. http://dx.doi.org/10.1007/978-3-642-79629-6_11

115. [Mataric 1995b] Mataric, M., Nilsson, M., & Simsarin, K. (1995). Cooperative multi-robot box-pushing. Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots. http://dx.doi.org/10.1109/iros.1995.525940

116. [Mataric 1997] Mataric, M. (1997). Behaviour-based control: examples from navigation, learning, and group behaviour. Journal of Experimental & Theoretical Artificial Intelligence, 9(2-3), 323-336. http://dx.doi.org/10.1080/095281397147149

117. [Mead 2009] Mead, R., Long, R., & Weinberg, J. (2009). Fault-tolerant formations of mobile robots. 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems. http://dx.doi.org/10.1109/iros.2009.5353996

118. [Meltzer 2004] Meltzer, J., Gupta, R., Ming-Hsuan Yang, & Soatto, S. (2004). Simultaneous localization and mapping using multiple view feature descriptors. 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566). http://dx.doi.org/10.1109/iros.2004.1389616

119. [Michael 2008] Michael, N., Zavlanos, M., Kumar, V., & Pappas, G. (2008). Distributed multi-robot task assignment and formation control. 2008 IEEE

International Conference on Robotics and Automation. http://dx.doi.org/10.1109/robot.2008.4543197

120. [Mondada 2009] Mondada F, Bonani M, Raemy X, Pugh J, et al (2009) The e-puck, a robot designed for education in engineering. Infoscience, EPFL's scientific publications. https://infoscience.epfl.ch/record/135236/files/epuck-robotica2009.pdf. Accessed 13 June 2016

121. [Monteiro 2010] Monteiro, S. & Bicho, E. (2010). Attractor dynamics approach to formation control: theory and application. Autonomous Robots, 29(3-4), 331-355. http://dx.doi.org/10.1007/s10514-010-9198-8

122. [Montemerlo 2003] Montemerlo, M. & Thrun, S. (2003). Simultaneous localization and mapping with unknown data association using FastSLAM. 2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422). http://dx.doi.org/10.1109/robot.2003.1241885

123. [Moravec 1985] Moravec, H. & Elfes, A. (1985). High resolution maps from wide angle sonar. Proceedings. 1985 IEEE International Conference on Robotics and Automation. http://dx.doi.org/10.1109/robot.1985.1087316

124. [Mukhija 2010] Mukhija, P., Krishna, K., & Krishna, V. (2010). A two phase recursive tree propagation based multi-robotic exploration framework with fixed base station constraint. 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems. http://dx.doi.org/10.1109/iros.2010.5649864

125. [Nunes 2015] Nunes, E., Gini, M. (2015). Multi-robot auctions for allocation of tasks with temporal constraints. In Proceedings, AAAI-2015

126. [Ozkan 2010] Ozkan, M., Kirlik, G., Parlaktuna, O., Yufka, A., & Yazici, A. (2010). A Multi-Robot Control Architecture for Fault-Tolerant Sensor-Based Coverage. Int J Adv Robotic Sy, 1. http://dx.doi.org/10.5772/7252

127. [Parker 1993] Parker, L. (1993). Designing control laws for cooperative agent teams. Proceedings. 1993 IEEE International Conference on Robotics and Automation. http://dx.doi.org/10.1109/robot.1993.291842

128. [Parker 1995] Parker, L. (1998). ALLIANCE: an architecture for fault tolerant multirobot cooperation. IEEE Trans. Robot. Automat., 14(2), 220-240. http://dx.doi.org/10.1109/70.681242

129. [Parrot ARdrone-2.0 2016] AR.Drone 2.0 official site – AR.Freeflight: Fly with iPhone and iPad. (2016). Parrot.com. Retrieved 23 August 2016, from http://www.parrot.com/usa/products/ardrone-2/

130. [Pini 2011] Pini, G., Brutschy, A., Frison, M., Roli, A., Dorigo, M., & Birattari, M. (2011). Task partitioning in swarms of robots: an adaptive method for strategy selection. Swarm Intell, 5(3-4), 283-304. http://dx.doi.org/10.1007/s11721-011-0060-1

131. [Portugal 2013] Portugal, D. & Rocha, R. (2013). Retrieving Topological Information for Mobile Robots Provided with Grid Maps. Communications in Computer and Information Science, 204-217. http://dx.doi.org/10.1007/978-3-642-36907-0_14

132. [Premvuti 1990] Premvuti, S. & Yuta, S. (1990). Consideration on the cooperation of multiple autonomous mobile robots. IEEE International Workshop on Intelligent Robots and Systems, Towards a New Frontier of Applications. http://dx.doi.org/10.1109/iros.1990.262369

133. [Prencipe 2001] Prencipe, G. (2001). CORDA: Distributed coordination of a set of autonomous mobile robots. 4th European Research Seminar on Advances in Distributed Systems (ERSADS 2001).

134. [Prencipe 2002] Prencipe, G. (2002). Distributed coordination of a set of autonomous mobile robots. Ph.D. Thesis, Università di Pisa

135. [Puig 2011] Puig, D., Garcia, M., & Wu, L. (2011). A new global optimization strategy for coordinated multi-robot exploration: Development and comparative evaluation. Robotics and Autonomous Systems, 59(9), 635-653. http://dx.doi.org/10.1016/j.robot.2011.05.004

136. [Radchuk 2013] Radchuk, D. (2013). Boost.Asio C++ Network Programming Cookbook. Packt Publishing.

137. [Raghavan 2007] Raghavan, S. & B, R. (2007). Homogeneous Hierarchical Composition of Areas in Multi-robot Area Coverage. Lecture Notes in Computer Science, 300-313. http://dx.doi.org/10.1007/978-3-540-73580-9_24

138. [Rekleitis 2001] Rekleitis, I., Dudek, G., & Milios, E. (2001). Annals of Mathematics and Artificial Intelligence, 31(1/4), 7-40. http://dx.doi.org/10.1023/a:1016636024246

139. [Rekleitis 2004] Rekleitis, I., Lee-Shue, V., Ai Peng New, & Choset, H. (2004). Limited communication, multi-robot team based coverage. IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004. http://dx.doi.org/10.1109/robot.2004.1308789

140. [Renzaglia 2010] Renzaglia, A. & Martinelli, A. (2010). Potential field based approach for coordinate exploration with a multi-robot team. 2010 IEEE Safety Security and Rescue Robotics. http://dx.doi.org/10.1109/ssrr.2010.5981557

141. [Reynolds 1987] Reynolds, C. (1987). Flocks, herds and schools: A distributed behavioral model. Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '87. http://dx.doi.org/10.1145/37401.37406

142. [Reynolds 1999] Reynolds C.W. (1999). Steering behaviors for autonomous characters. In Proceedings of Game Developers Conference, San Francisco, CA

143. [Reynolds 2000] Reynolds C.W. (2000). Interaction with a group of autonomous characters. In Proceedings of Game Developers Conference, San Francisco, CA

144. [Rhoad 1984] Rhoad, R. (1984). Tests for Geometry for enjoyment and challenge. Evanston, Ill.: McDougal, Littell.

145. [Rooker 2007] Rooker, M. & Birk, A. (2007). Multi-robot exploration under the constraints of wireless networking. Control Engineering Practice, 15(4), 435-445. http://dx.doi.org/10.1016/j.conengprac.2006.08.007

146. [Russell 1995] Russell, S. & Norvig, P. (1995). Artificial intelligence. Englewood Cliffs, N.J.: Prentice Hall.

147. [Schneider-Fontan 1998] Schneider-Fontan, M. & Mataric, M. (1998). Territorial multi-robot task division. IEEE Trans. Robot. Automat., 14(5), 815-822. http://dx.doi.org/10.1109/70.720357

148. [SciStatCalc 2016] SciStatCalc: Wilcoxon Signed Rank Test Calculator. (2016). Scistatcalc.blogspot.in. Retrieved 21 August 2016, from http://scistatcalc.blogspot.in/2013/10/wilcoxon-signed-rank-test-calculator.html

149. [Senthilkumar 2012] Senthilkumar, K. & Bharadwaj, K. (2012). Multi-robot exploration and terrain coverage in an unknown environment. Robotics and Autonomous Systems, 60(1), 123-132. http://dx.doi.org/10.1016/j.robot.2011.09.005

150. [Sharpe 1998] Sharpe, R. & Webb, B. (1998). Simulated and situated models of chemical trail following in ants. 1998 International Conference on Simulation of Adaptive Behavior.

151. [Shell 2006] Shell, D. & Mataric, M. (2006). On foraging strategies for large-scale multi-robot systems. 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems. http://dx.doi.org/10.1109/iros.2006.281996

152. [Sheng 2006] Sheng, W., Yang, Q., Tan, J., & Xi, N. (2006). Distributed multi-robot coordination in area exploration. Robotics and Autonomous Systems, 54(12), 945-955. http://dx.doi.org/10.1016/j.robot.2006.06.003

153. [Sit 2007] Sit, T., Liu, Z., Ang Jr., M., & Seah, W. (2007). Multi-robot mobility enhanced hop-count based localization in ad hoc networks. Robotics and Autonomous Systems, 55(3), 244-252. http://dx.doi.org/10.1016/j.robot.2006.08.005

154. [Skyum 1991] Skyum, S. (1991). A simple algorithm for computing the smallest enclosing circle. Information Processing Letters, 37(3), 121-125. http://dx.doi.org/10.1016/0020-0190(91)90030-l

155. [Solanas 2004] Solanas, A. & Garcia, M. (2004). Coordinated multi-robot exploration through unsupervised clustering of unknown space. 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566). http://dx.doi.org/10.1109/iros.2004.1389437

156. [Souissi 2004] Souissi, S., Défago, X., Katayama, T. (2004). Convergence of a Self-Stabilizing Circle Formation Algorithm for Cooperative Mobile Robotics. In Proceedings of Scientific French-speaking Workshops (JSF'04). Tokyo, Japan.

157. [Stachniss 2006] Stachniss, C. (2006). Exploration and Mapping with Mobile Robots. PhD thesis, University of Freiburg, Department of Computer Science, April 2006

158. [Sugihara 1996] Sugihara, K. & Suzuki, I. (1996). Distributed algorithms for formation of geometric patterns with many mobile robots. J. Robotic Syst., 13(3), 127-139. http://dx.doi.org/10.1002/(sici)1097-4563(199603)13:3<127::aid-rob1>3.0.co;2-u

159. [Suzuki 1999] Suzuki, I. & Yamashita, M. (1999). Distributed Anonymous Mobile Robots: Formation of Geometric Patterns. SIAM J. Comput., 28(4), 1347-1363. http://dx.doi.org/10.1137/s009753979628292x

160. [Svennebring 2004] Svennebring, J. & Koenig, S. (2004). Building Terrain-Covering Ant Robots: A Feasibility Study. Autonomous Robots, 16(3), 313-332. http://dx.doi.org/10.1023/b:auro.0000025793.46961.f6

161. [Tanner 2003] Tanner, H., Loizou, S., & Kyriakopoulos, K. (2003). Nonholonomic navigation and control of cooperating mobile manipulators. IEEE Trans. Robot. Automat., 19(1), 53-64. http://dx.doi.org/10.1109/tra.2002.807549

162. [Thrun 1998] Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation. Artificial Intelligence, 99(1), 21-71. http://dx.doi.org/10.1016/s0004-3702(97)00078-7

163. [Trebi-Ollennu 1999] Trebi-Ollennu, A., & Dolan, J. M. (1999). An autonomous ground vehicle for distributed surveillance: cyberscout. Technical Report ICES-04-09-99. School of Computer Science, Carnegie Mellon University.

164. [Turtlebot 2016] TurtleBot. (2016). Turtlebot.com. Retrieved 23 August 2016, from http://www.turtlebot.com/

165. [Vasisht 2016] Vasisht, D., Kumar, S., & Katabi, D. (2016). Decimeter-level localization with a single WiFi access point. 13th USENIX Symposium on Networked Systems Design and Implementation

166. [Viguria 2007] Viguria, A., Maza, I., & Ollero, A. (2007). SET: An algorithm for distributed multirobot task allocation with dynamic negotiation based on task subsets. Proceedings 2007 IEEE International Conference on Robotics and Automation. http://dx.doi.org/10.1109/robot.2007.363988

167. [Viguria 2008] Viguria, A., Maza, I., & Ollero, A. (2008). S+T: An algorithm for distributed multirobot task allocation based on services for improving robot cooperation. 2008 IEEE International Conference on Robotics and Automation. http://dx.doi.org/10.1109/robot.2008.4543692

168. [Wang 1991] Wang, P. (1991). Navigation strategies for multiple autonomous mobile robots moving in formation. J. Robotic Syst., 8(2), 177-195. http://dx.doi.org/10.1002/rob.4620080204

169. [Wang 2005] Wang, Z., Liu, L. & Zhou, M. (2005). Protocols and Applications of Ad-hoc Robot Wireless Communication Networks: An Overview. International Journal of Intelligent Control Systems, 10(4), 296- 303

170. [Weitzenfeld 2006a] Weitzenfeld, A., Vallesa, A., & Flores, H. (2006). A Biologically-Inspired Wolf Pack Multiple Robot Hunting Model. 2006 IEEE 3rd Latin American Robotics Symposium. http://dx.doi.org/10.1109/lars.2006.334327

171. [Weitzenfeld 2006b] Weitzenfeld, A., Martinez-Gomez, L., Francois, J., Levin-Pick, A., Obraczka, K., & Boice, J. (2006). Multi-Robot Systems: Extending RoboCup Small-Size Architecture with Local Vision and Ad-Hoc Networking. 2006 IEEE 3rd Latin American Robotics Symposium. http://dx.doi.org/10.1109/lars.2006.334328

172. [Witkowski 2008] Witkowski, U., El-Habbal, M., Herbrechtsmeier, S., Tanoto, A., Penders, J., Alboul, L. & Gazi, V. (2008). Ad-hoc network communication infrastructure for multi-robot systems in disaster scenarios. International Workshop on Robotics for Risky Interventions and Surveillance of the Environment

173. [Wu 2007] Wu, L., García García, M., Puig Valls, D., & Solé Ribalta, A. (2007). Voronoi-based space partitioning for coordinated multi-robot exploration. Journal of Physical Agents (Jopha), 1(1), 37-44. http://dx.doi.org/10.14198/jopha.2007.1.1.05

174. [Wu 2010] Wu, L., Puig Valls, D., & García García, M. (2010). Balanced multi-robot exploration through a global optimization strategy. Red De Agentes Físicos. Retrieved from http://dx.doi.org/10.14198/JoPha.2010.4.1.06

175. [Wurm 2008] Wurm, K., Stachniss, C., & Burgard, W. (2008). Coordinated multi-robot exploration using a segmentation of the environment. 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems. http://dx.doi.org/10.1109/iros.2008.4650734

176. [Xiong 2010] Xiong, N., He, J., Yang, Y., He, Y., Kim, T., & Lin, C. (2010). A Survey on Decentralized Flocking Schemes for a Set of Autonomous Mobile Robots (Invited Paper). Journal of Communications, 5(1). http://dx.doi.org/10.4304/jcm.5.1.31-38

177. [Yamada 2001] Yamada, S. & Saito, J. (2001). Adaptive action selection without explicit communication for multirobot box-pushing. IEEE Trans. Syst., Man, Cybern. C, 31(3), 398-404. http://dx.doi.org/10.1109/5326.971668

178. [Yamashita 2003] Yamashita, A., Arai, T., Jun Ota, & Asama, H. (2003). Motion planning of multiple mobile robots for cooperative manipulation and transportation. IEEE Trans. Robot. Automat., 19(2), 223-237. http://dx.doi.org/10.1109/tra.2003.809592

179. [Yamauchi 1997] Yamauchi, B. (1997). A frontier-based approach for autonomous exploration. 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'. http://dx.doi.org/10.1109/cira.1997.613851

180. [Yamauchi 1998] Yamauchi, B. (1998). Frontier-based exploration using multiple robots. 2nd International Conference on Autonomous Agents - AGENTS '98. http://dx.doi.org/10.1145/280765.280773

181. [Yan 2013] Yan, Z., Jouandeau, N., & Ali, A. (2013). A Survey and Analysis of Multi-Robot Coordination. Int J Adv Robotic Sy, 1. http://dx.doi.org/10.5772/57313

182. [Yang 2009] Yongming Yang, Changjiu Zhou, & Yantao Tian. (2009). Swarm robots task allocation based on response threshold model. 2009 4th International Conference on Autonomous Robots and Agents. http://dx.doi.org/10.1109/icara.2000.4803959

183. [Yang 2011] Yang, Y., Souissi, S., Défago, X., & Takizawa, M. (2011). Fault-tolerant flocking for a group of autonomous mobile robots. Journal of Systems and Software, 84(1), 29-36. http://dx.doi.org/10.1016/j.jss.2010.08.026

184. [Yazıcıoğlu 2013] Yazıcıoğlu, A., Egerstedt, M., & Shamma, J. (2013). A Game Theoretic Approach to Distributed Coverage of Graphs by Heterogeneous Mobile Agents. IFAC Proceedings Volumes, 46(27), 309-315. http://dx.doi.org/10.3182/20130925-2-de-4044.00034

185. [Zavlanos 2008] Zavlanos, M., Spesivtsev, L., & Pappas, G. (2008). A distributed auction algorithm for the assignment problem. 2008 47th IEEE Conference on Decision and Control. http://dx.doi.org/10.1109/cdc.2008.4739098

186. [Zheng 2008] Taixiong Zheng, & Liangyi Yang. (2008). Optimal ant colony algorithm based multi-robot task allocation and processing sequence scheduling. 2008 7th World Congress on Intelligent Control and Automation. http://dx.doi.org/10.1109/wcica.2008.4593859

187. [Zheng 2010] Zheng, X., Koenig, S., Kempe, D., & Jain, S. (2010). Multirobot Forest Coverage for Weighted and Unweighted Terrain. IEEE Trans. Robot., 26(6), 1018-1031. http://dx.doi.org/10.1109/tro.2010.2072271

188. [Xu 2009] Xu, Z., Xia, F., & Zhang, X. (2009). Multi-Robot Dynamic Task Allocation Using Modified Ant Colony System. Artificial Intelligence and Computational Intelligence, 288-297. http://dx.doi.org/10.1007/978-3-642-05253-8_32

189. [ZhiDong 2003] ZhiDong Wang, Nakano, E., & Takahashi, T. (2003). Solving function distribution and behavior design problem for cooperative object handling by multiple mobile robots. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 33(5), 537-549. http://dx.doi.org/10.1109/tsmca.2003.817396

190. [Zlot 2002] Zlot, R., Stentz, A., Dias, M., & Thayer, S. (2002). Multi-robot exploration controlled by a market economy. 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292). http://dx.doi.org/10.1109/robot.2002.1013690

APPENDIX

The appendix is attached as a CD containing three videos of the experiments we have conducted in simulation and on a multi-robot test-bed in a laboratory setting. Following is the description of each of the video files:

(a) **VID-1_STATE.mp4 –** This video demonstrates the execution of the proposed STATE algorithm in Chapter 3 of the thesis. Initially the robots move to some random positions on the floor. All robots find the smallest enclosing circle (SEC) in their own local coordinate system. The center of the SEC coincides and the robots transform their local reference frame to a global reference frame while considering the center of the SEC as the origin. The robots then move to the circumference of the SEC. Once all the robots are positioned on the SEC, each robot locally starts executing the STATE algorithm. It can be seen that the team of five e-puck robots eventually make a uniform circular formation.

(b) **VID-2_CAC.mp4 –** This video demonstrates the execution of the proposed CAC algorithm in Chapter 4 of the thesis. The robots start from a common entry point and starts covering the terrain. The emergent dispersion of the robots is visible in the video. It can be noticed in the beginning that the robot with the red trail deliberately leaves one cell uncovered in the lower left-hand corner of the terrain. This cell is marked as a home cell. In the end when the entire terrain is covered and only the home cell is left all the robots return to home.

(c) **VID-3_FAST.mp4 –** This video demonstrates the execution of the proposed FAST algorithm in Chapter 5 of the thesis. The FAST algorithm has a property that it creates large unbroken segments while covering the terrain. This property is demonstrated in simulation. Besides FAST, the execution of two other state-of-the-art algorithms in simulation is also shown. It can be observed that the other two algorithms cover the terrain in a haphazard manner. Subsequently, the execution of the algorithm FAST on an experimental test-bed is demonstrated. Three robots are shown to cover the terrain without failing. Later, three robots cover the terrain while one robot fails after every six steps (it was stopped intentionally). It can be seen that the last robot with the blue trail covers remaining terrain and completes the terrain coverage task.

LIST OF PUBLICATIONS

JOURNAL PUBLICATIONS

1. Gautam, A. & Mohan, S. (2016). STATE: Distributed Algorithm for Uniform Circle Formation by Multiple Mobile Robots. **Journal of Intelligent Service Robotics**, 9(4), 347-366. DOI: 10.1007/s11370-016-0205-6. Available from http://rdcu.be/jZgI. Indexing: **SCI-Expanded**

2. Gautam, A., Jha, B., Kumar, G., Murthy, J. K., Ram, S. P. A. & Mohan, S. (2016). FAST: Synchronous Frontier Allocation for Scalable Online Multi-Robot Terrain Coverage. **Journal of Intelligent Robotic Systems (Springer)**. DOI: 10.1007/s10846-016-0416-2. Available from http://rdcu.be/kpnE. Indexing: **SCI-Expanded**

CONFERENCE PAPERS

1. Gautam, A. & Mohan, S. (2012). A review of research in multi-robot systems. In Proceedings of *2012 IEEE 7th International Conference on Industrial and Information Systems (**ICIIS 2012**)*. DOI: 10.1109/ICIInfS.2012.6304778. Location: **IIT Madras, Chennai, India.** Indexing: **Scopus**

2. Gautam, A., Mohan, S., & Misra, J. (2012). A practical framework for uniform circle formation by multiple mobile robots. In Proceedings of *2012 IEEE 7th International Conference on Industrial and Information Systems (**ICIIS 2012**)*. DOI: 10.1109/iciinfs.2012.6304765. Location: **IIT Madras, Chennai, India.** Indexing: **Scopus**

3. Gautam, A., Mohan, S., & Shekhawat, V. (2013). A token passing approach for circle formation by multiple mobile robots. In Proceedings of *2013 6th International Conference on Contemporary Computing (**IC3 2013**)*. DOI: 10.1109/ic3.2013.6612251. Location: **JIIT University, Noida, India.** Indexing: **Scopus**

4. Gautam, A. & Mohan, S. (2013). A distributed algorithm for circle formation by multiple mobile robots. In Proceedings of *2013 International Conference on Control, Automation, Robotics and Embedded Systems (**CARE 2013**)*. DOI: 10.1109/care.2013.6733699. Location: **IIITDM, Jabalpur, India.** Indexing: **Scopus**

5. Gautam, A., Saxena, A., Mall, P., & Mohan, S. (2014). Positioning multiple mobile robots for geometric pattern formation: An empirical analysis. In Proceedings of *2014 7th International Conference on Contemporary Computing (**IC3 2014**)*. DOI: 10.1109/ic3.2014.6897242. Location: **JIIT University, Noida, India.** Indexing: **Scopus**

6. Gautam, A., Murthy, J., Kumar, G., Ram, S., Jha, B., & Mohan, S. (2015). Cluster, Allocate, Cover: An Efficient Approach for Multi-Robot Coverage. In Proceedings of *2015 IEEE International Conference on Systems, Man, and Cybernetics* (**SMC 2015**). DOI: 10.1109/smc.2015.47. Location: **City University, Hong Kong, China.** Indexing: **Scopus**

7. [**Accepted Manuscript**] Gautam, A., Thakur, A., Dhanania, G., & Mohan, S. (2016). A Distributed Algorithm for Balanced Multi-Robot Task Allocation. In *2016 IEEE 11th International Conference on Industrial and Information Systems (**ICIIS 2016**)*. **To be held at IIT Roorkee, India, from 3-4 December 2016.**

JOURNAL PAPERS (UNDER REVIEW)

1. Gautam, A., Ram, S. P. A. & Mohan, S. (2016). Balanced Partitioning of an Unknown Region for Efficient Multi-Robot Coordination. [**Manuscript submitted for publication**]

BRIEF BIOGRAPHY OF CANDIDATE

Mr. Avinash Gautam is a research scholar in the Department of Computer Science & Information Systems at Birla Institute of Technology and Science, Pilani (Pilani campus) since December 2009. He is also working as a Lecturer at BITS, Pilani. He obtained his B.E. (Hons) degree in Computer Engineering from University of Rajasthan, Jaipur, and M.E. degree in Software Systems from BITS Pilani (Pilani campus), in 2005 & 2008 respectively. His research interests include cooperative robotics and its applications, multi-agent systems, and distributed computing.

BRIEF BIOGRAPHY OF SUPERVISOR

Prof. S. Mohan completed his PhD in Electrical Engineering from the Birla Institute of Technology and Sciences (BITS), Pilani. He also holds a Masters (MSc.) degree in Physics and Master of Engineering (M.E) in Electronics and Control from the same institute. He has a teaching and research experience of over twenty-five years at BITS, Pilani. Previously he has served as Head of the Computer Science and Information Systems department and Dean of Admissions at BITS Pilani. Currently he is Professor of Computer Science department at BITS Pilani (Pilani campus). His research interests include Automatic Controls and Robotics.