DYNAMIC FORMATION CONTROL WITH HETEROGENEOUS MOBILE ROBOTS

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

BY

KADİR ÇİMENCİ

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

JUNE 2016

Approval of the thesis:

DYNAMIC FORMATION CONTROL WITH HETEROGENEOUS MOBILE ROBOTS

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

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

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

Prof. Dr. Aydan Müşerref Erkmen
Supervisor, **Electrical and Electronics Engineering Department** _____

Examining Committee Members:

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

Assoc. Prof. Dr. Umut Örgüner
Electrical and Electronics Engineering Department, METU _____

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

Assoc. Prof. Dr. Yiğit Yazıcıoğlu
Mechanical Engineering Department, METU _____

Assist. Prof. Dr. Elif Vural
Electrical and Electronics Engineering Department, METU _____

Date: _____

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

Name, Last Name: KADİR ÇİMENCİ

Signature :

ABSTRACT

DYNAMIC FORMATION CONTROL WITH HETEROGENEOUS MOBILE ROBOTS

ÇİMENCİ, KADİR

M.S., Department of Electrical and Electronics Engineering

Supervisor : Prof. Dr. Aydan Müşerref Erkmen

June 2016, 153 pages

Formation control in robotics is a growing topic where research works are mainly geared towards heterogeneous swarm colonies under either decentralized control or limited centralization. Swarm robotics where decentralization is applied, nevertheless assume that the agents are capable of getting global information about the whole swarm.Moreover in the literature, formation control is generally done for known fixed shapes that can be defined mathematically. However no dynamically changing shapes are envisaged and no shape transitions are clearly handled in those works. We attempt to bring a clear impact to the literature by focusing on tracking and realising formation shapes under dynamically changing formation shape demands. Furthermore,in our thesis work, we focus on robot colonies composed of heterogeneous robots of different dynamics and sensor capabilities under decentralized dynamically changing formation control. These robots are able furthermore, to just possess local mutual interactions only with their close-by neighboring agents. In our approach communications of each agent with its neighbors converges to information about the whole colony based on graph theory. Simulations in our work are generated using the Gazebo environment by modelling a rough territory. Hardware applications which implements the methods discussed in this thesis work are also developed. These applications are evaluated as proof of concept work which illustrates that the methods can be implemented in real time applications.

Keywords: Multi Agent Systems, Formation Control, Localization, Mesh Network

ÖZ

HETEROJEN ROBOTLARLA DİNAMİK FORMASYON KONTROLÜ

ÇİMENCİ, KADİR
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Aydan Müşerref Erkmen

Haziran 2016 , 153 sayfa

Formasyon kontrolü robotik alanında heterojen robot kolonilerinin merkezi yönetici birimler olmadan ya da yerel merkezi birimleri barındıran sürülere yönelik büyüyen bir araştırma alanıdır. Merkezi yönetim birimlerinin olmadığı robot sürüsü çalışmalarında ne yazık ki her üyenin, koloniye mensup diğer üyelerin tüm verilerine erişebildiği varsayımı yapılmaktadır.Öte yandan, literatürde formasyon kontrolü genellikle matematiksel olarak ifade edilebilen geometrik şekillerle yapılmaktadır. Bununla birlikte bu çalışmalarda, dinamik olarak değişen şekiller ve bu şekiller arasında formasyon geçişleri konusu yeterli olarak ele alınmamaktadır. Bu çalışma kapsamında dinamik olarak değişen şekiller için formasyon kontrolü sağlayarak literatüre katkıda bulunmayı hedefledik. Öte yandan bu tez çalışmamızda, farklı dinamiklere ve sensör yetkinliklerine sahip heterojen robot kolonileri kullanarak dinamik formasyon kontrolü problemini merkezi karar verici birimlerin olmadığı bir topolojide ele aldık. Çalışma kapsamında ele alınan kolonilerdeki tüm robotlar yalnızca kendilerine en yakın komşu üyelerle yerel etkileşimlerde bulunabilmektedir. Komşularıyla etkileşimde bulunan üyeler, koloninin geri kalanı hakkında bilgi sahibi olabilmektedir. Çalışmamız kapsamında simülasyonlar Gazebo ortamında üç boyutlu düzgün olmayan araziler modellenerek yapılmıştır. Ayrıca donanımsal gerçeklemeler içeren çalışmalar da yapılmıştır. Bu çalışmalar, tez kapsamında önerilen yöntemlerin gerçek zamanlı sistemlerde gerçeklenebileceğenin bir göstergesi olarak ele alınmaktadır.

Anahtar Kelimeler: Çok Elemanlı Sistemler, Formasyon Kontrolü, Konumlama, Örgün Ağlar

To my family and people who are reading this page

ACKNOWLEDGMENTS

I would like to express my gratitude to my supervisor Prof. Dr. Aydan ERKMEN for the useful comments, remarks and engagement through the learning process of this master thesis. Furthermore I would like to thank TUBITAK for financial support during the time of my graduate school education. Also, I would like to thank my loved ones, my family and my friends, who have supported me throughout entire process, both by keeping me harmonious and helping me putting pieces together. I will be grateful forever for your love.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ALGORITHMS

ALGORITHMS

CHAPTER 1

INTRODUCTION

This thesis work focuses on dynamic adaptation to achieve changes in formation of swarms consisting of heterogeneous mobile robots. The term of swarm has a meaning [1], a large group of locally interacting individuals with common goals.

Self organizing swarm researches and its applications are generally inspired from the biological systems existing in the nature. Behaviours of complex biological systems remained unresolved for a long time, until recent researches that showed individuals with supple functionalities can achieve complex tasks by cooperation [2]. Biological systems (such as colonies of ants) have simple behaviours but they can accomplish very complicated collective tasks in nature quite impossible to be achieved solely by individual capabilities. Beni [3] describes this collaboration of members as follows:

Collaborative robot network is not just a agglomeration of robots, they have essential characteristics, which are found in swarms of insects, that is, decentralised control, which does not require synchronisation, and individuals which are simple and (quasi) identical.

Properties of swarm robotics systems that make them unique are the simplicity of individuals, restricted sensing and communication capabilities, collaborative achievement of complex tasks, robustness and decentralization of control capabilities [4].

Swarm robotics has been studied to produce different collective behaviours to solve tasks such as as aggregation , pattern formation , self-assembly and morphogenesis , object clustering, assembling and construction , collective search&rescue and exploration , coordinated motion , collective transportation , self-deployment , foraging

and others[5]. Dorigo and Trianni [6] are studied on controllers for aggregation of coordinated motion of the identical mobile robots called swarm-bots. Their implementation has a centralized structure in which a decision maker assigns roles to individual agents. In our work, each agent is responsible to reach global consensus on the final positions in formation which increase the robustness of the system against to individual erroneous decisions. Hou et al. is focused on controlling of a swarm within a dynamically changing formation [7]. In their work, it is needed to describe a global objective function which requires analytical expression of the desired formation shape. But it may not be possible to define the desired formation shape with analytical expressions in real world applications. In this thesis work, the solution does not require the analytical expressions of the desired formation shape. Ganesh and Lisa introduced two new strategies for collective search and exploration of fields with swarm intelligence [8]. Their implementation requires a central controller to manage the individuals and this type of solution has a single point of failure system characteristic because of the dependence on single decision maker agent. Chaimowicz and Campos proposed a new methodology which is based on a dynamic role assignment mechanism in which the robots cooperate with each other and they demonstrate this method in a cooperative transportation task [9]. Their work implements a leader follower structure in which following agents has no impact on the global consensus of the swarm. This approach depends on the decision of an individual agent and an erroneous decision of this individual may cause the failure of whole swarm.

There are lots of studies related with different problems in swarm robotics literature as discussed briefly. In this thesis project, we are focused on dynamic pattern formation control of swarms consist of heterogeneous robots. All agents contribute on the decision process and they collaborate to minimize the total displacement while achieving the desired formation shape.

1.1 Problem Definition

In this thesis work, it is aimed to design a formation control system with heterogeneous robots with dynamically changing complex shapes. Desired formation shapes for real time applications, such as covering an area or a search&rescue operation, may

not be simple geometrical shapes with analytical expressions. The solution which does not require a mathematical definition of the desired formation shape is more suitable for these type of applications. On the other hand, this formation shape may be dynamically changing to meet the requirements of the task in real time.

In literature, formation control systems generally include a decision making process which is executed by an individual agent or a central server. This kind of approach creates a single point of failure system and this ruler may cause a failure for the whole swarm by making a mistake. In this work, it is aimed to implement a solution in which each member of the swarm contributes on the decisions and global consensus of the swarm.

Formation control systems which are implemented with swarms composed of homogeneous agents cannot achieve special tasks which requires different functionalities of individual agents. In this work, we have focused on implementing a solution with a swarm consisting of heterogeneous agents with different capabilities.

In this work, the main idea is to propose a complete design solution to a dynamically changing formation system, including a local positioning and formation control system. Formation control system heavily depends on the position data of individual agents in the workspace. Since it is expected to have high number of agents in the environment due to the nature of a swarm, the agents are assumed to have simple structures with low capabilities including lack of certain types of sensors. On the other hand, for indoor applications it will not be possible to use satellite dependent positioning systems on agents [10]. Even if a positioning solution provided for an indoor application by different methods including visual feedback(by image processing), RSS(received signal strength) etc. , it is not possible to implement this solution for each single agent in the environment due to the increasing complexity and the costs by the number of agents. In literature, formation control systems are generally designed with the assumption of the positions of each agent is known exactly. To propose a complete solution for the formation control problem, it is required to implement a localization solution for the agents to provide the corrected positions in the workspace which will be used by the formation control system. This localization process must implement a solution to correct the position data of the agents with the

determined process period and within an maximum error bound. This error bound is determined by the requirements of the formation control problem.

1.2 Motivation

The formation control problem is defined as the collaboration of a group of agents towards maintaining a formation with a certain shape [11]. It focuses on leading the individual agents of a swarm to perform collective tasks including shape generation and formation reconfiguration while traversing a trajectory avoiding collisions. These kind of tasks are achieved with a large group of small and simple robots that can cooperate with each other. Formation control of multi agent systems is an actively growing research field.

Swarms which are used in formation control systems, can be composed of homogeneous or heterogeneous agents according to the requirements of the problem. The usage of the homogeneous agents increases the total impact and the coverage of the system in the environment. This kind of a swarm has an increased redundancy and is capable of resuming the current task in case of failures of some of the agents during mission. On the other hand, a swarm composed of heterogeneous agents has different capabilities of the agents. This kind of a system can be used in tasks which requires different functionalities has to be performed individually or simultaneously.

In real world applications there may be need for different functionalities to achieve some specific tasks. If this is the case, one solution may be to design a sophisticated robot which includes all required capabilities for this task. In this scenario, this robot will be the single point of failure in the system and if robustness is a vital feature for this solution, some redundant robots have to be added to the system. It is clear that the design of such an advanced robot and hold its redundant backups in the system will increase the cost of the solution. In swarm robotics concept, one of the approaches related with the usage of the heterogeneous agents is to gather some different types of simple mobile robots which have their own specific functionalities to achieve a collective task rather than designing an advanced robot for the solution [12]. With this approach, the robustness of the system is increased, costs are reduced down and

the reusability of the individual members of the swarm for other tasks is provided. A project named Swarmanoid which is funded by European Commission, has an objective to implement and control of a novel distributed robotic system. The system is designed with heterogeneous, dynamically connected, small autonomous robots called foot-bots , hand-bots and eye-bots where foot-bots are responsible to transport the required materials(including other types of robots) to a specific task area and hand-bots are responsible for operations to be handled with their manipulators and eye-bots are responsible of observations and reconnaissance on the workspace. This project implements a system which has a leader follower structure in which a decision maker agent assign different tasks and roles to the other members of the swarm. Wrong decisions or the failure of this decision maker may prevent the whole swarm to achieve the desired task. In this thesis work, we have focused on a system in which each agent is responsible to contribute on the decision process.

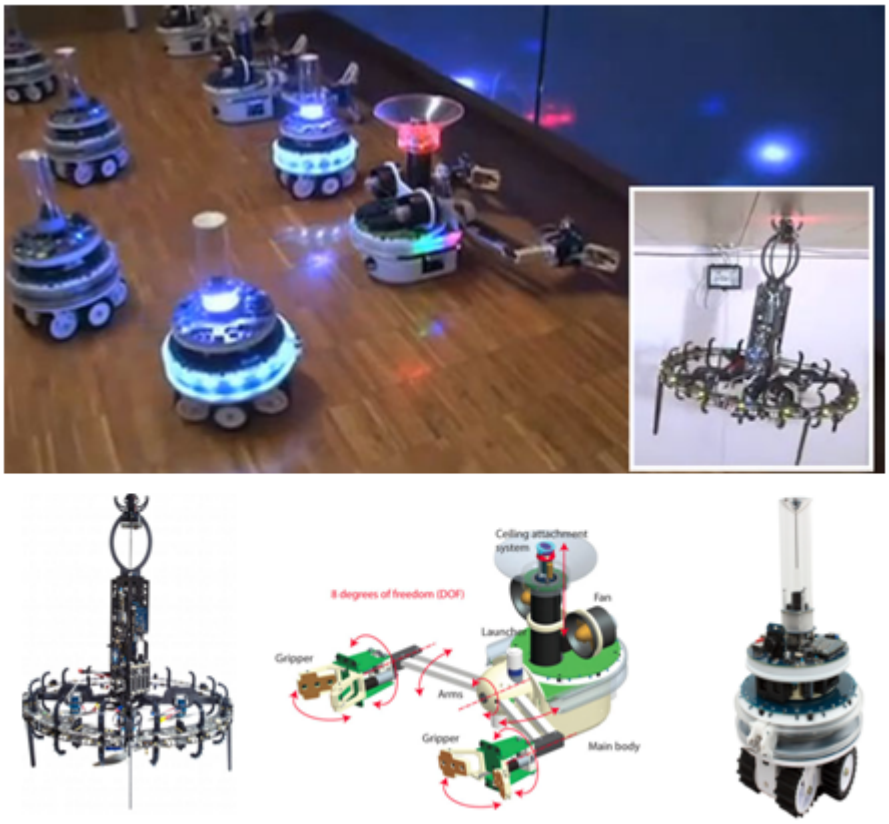


Figure 1.1: A Robot Team Consists of Eyebot, Handbot and Footbot Agents [12]

A swarm which is composed with same type of homogeneous agents can be used to

increase the total impact and the energy of an individual agent. This kind of a system can be used in missions like coverage, search and reconnaissance etc. Martin and Kilberg have worked on formation control and formation tracking of microsatellites to achieve continuous coverage and improved capability. They also mentioned that small formations will reduce the fuel consumption for propulsion and expand the sensing capabilities of microsatellites [13].

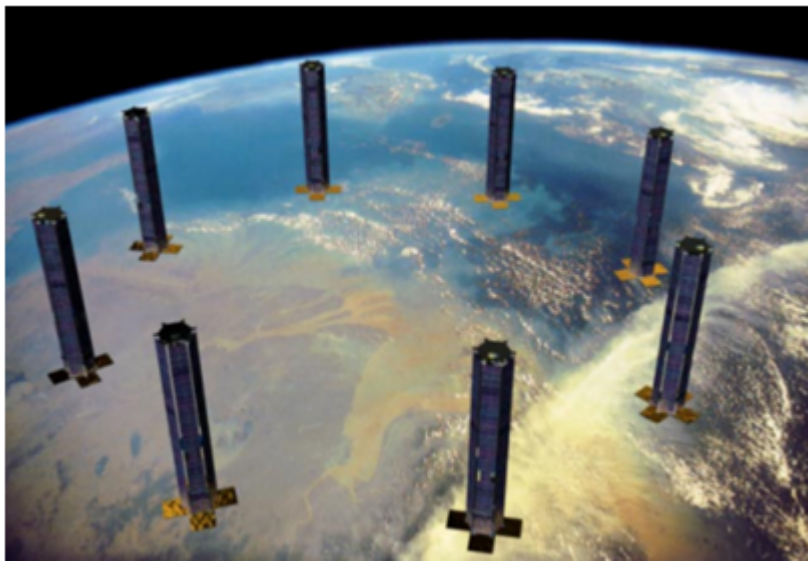


Figure 1.2: Sparse Aperture Formation of Micro Satellites [13]

Formation control solutions has lots of usage areas such as coverage missions, security patrols and search&rescue in hazardous environments etc.[14]. For missions related with area coverage and reconnaissance, a group of autonomous vehicles may be required to keep in a specified formation [14].

There are some hardware implementations to test the related formation control algorithms in real time applications. Since the formation control problem requires lots of agents in a swarm, these works have a common point of providing agents with minimal costs and sensor capabilities. The Kilobot Project from Harvard university have released their agents with the name of Kilobots and they have teams which are working on different formation control problems with Kilobots [15].

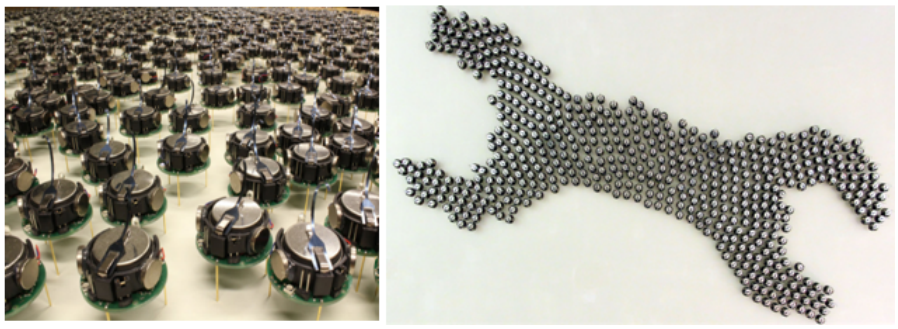


Figure 1.3: Formation Control with Kilobots [15]

These micro robots have a great reusability for different types of formation control problems and they have biological inspiration from the nature in the sense of individual simplicity and power of collective behaviours. Common feature of these projects, they implement formation control systems with swarms composed of homogeneous agents. In real world applications, complex tasks may require different functionalities which can be provided by different types of robots. In our work, we have focused on implementing a formation control system with heterogeneous agents.

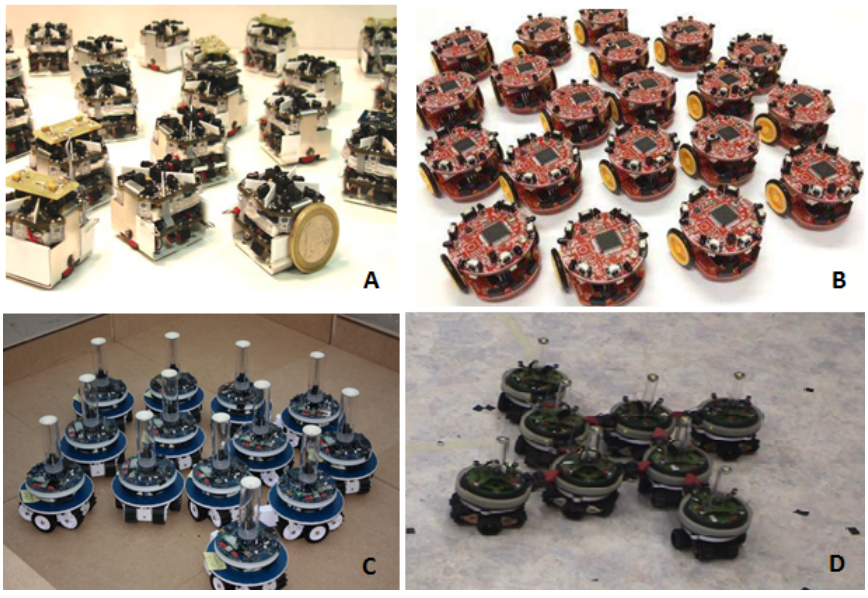


Figure 1.4: A) Swarm Robot Project from Universities of Stuttgart [16]
B) Colias Project from Lincoln University and Tsinghua University[17]
C) Marx bot developed at EPFL[18]
D)Swarm bots project conducted by European Commission [19]

Hou et al. has implemented a formation control system with dynamically changing formation shapes [7]. Their approach defines a global objective function which requires the mathematical definition of the desired formation shape. But in real world applications it may not be possible to derive the analytical expressions for the desired shapes. In our work, control laws for individuals in the swarm does not require such a global objective function definition.

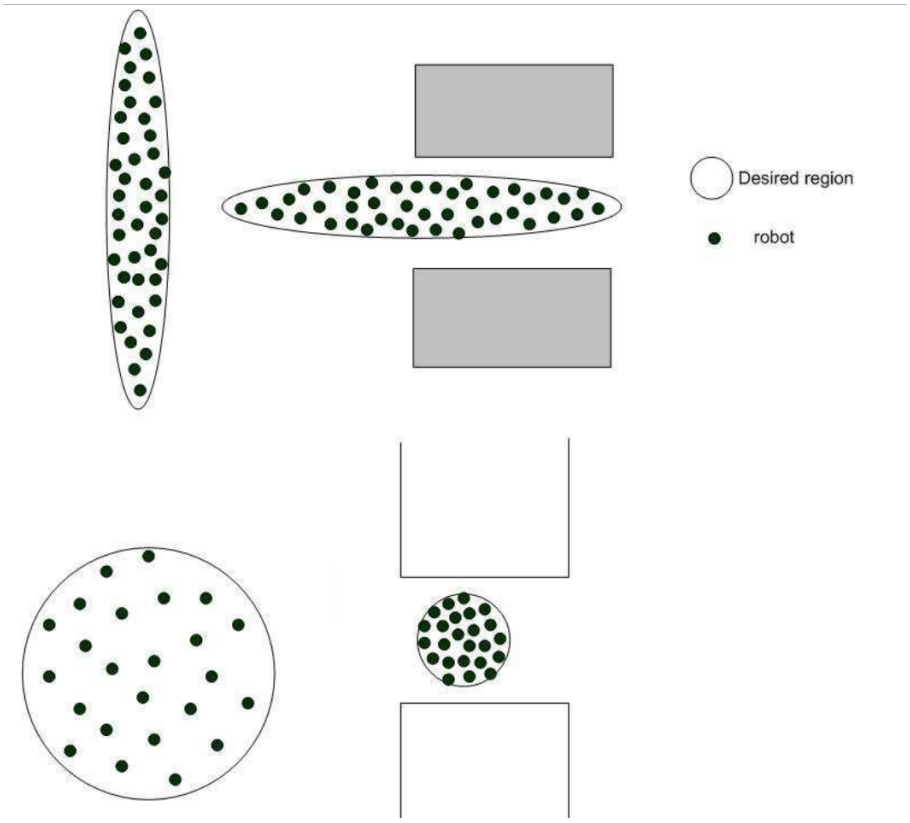


Figure 1.5: Dynamical Formations with Global Objective Functions [7]

1.3 Objectives & Goals

In this thesis work, our aim is to provide different approaches & solutions to the requirements in formation control problem. There are mainly different types of infrastructures which propose solutions to the formation control problems like the heterogeneity vs. homogeneity of the agents, communication structures, centralized vs. decentralized structures, swarm control strategies like behaviour based and leader-following approaches or virtual structure based approaches.

Our main approach is to design a complete solution to the formation control problem by taking into account the requirements related with real world applications. In literature, formation control systems generally implements solutions with simple geometrical shapes [7]. One of the objectives of this thesis work is to implement a solution which does not require the mathematical definition of the desired formation shape. Also it is aimed to design an algorithm which can adapt itself to the dynamically changing formations.

On the other hand, most of the related work assumes that it is possible to measure the exact positions of the agents in the workspace [15, 16]. Another objective of this thesis work is to implement a localization solution for the agents by assuming they may not able to measure their positions. With this approach, it is possible to provide a complete solution to the formation control problem.

Specific tasks including different missions, requires agents with different capabilities and this kind of tasks may not be achieved with swarms composed of homogeneous agents [12]. In our work, one of the objectives is to implement a formation control system with heterogeneous agents. Furthermore, proposed solutions for formation control problem generally includes a decision making process which is executed by an individual agent or a central server. This kind of approach creates a single point of failure system and if this decision maker unit fails during mission, swarm cannot achieve the desired task. In this thesis work, it is aimed to implement a solution in which every agent is responsible of contributing on decisions and reaching a global consensus.

In addition to choice of the formation control infrastructures, capabilities of the individual agents provide additional requirements and constraints while designing a formation control system. One of the most important characteristic of an agent in the swarm is its simplicity and limited sensor & communication capability [4]. This approach results from the idea of achieving collective tasks with lots of simple individuals and it is based on biological inspirations in nature like colonies of ants etc. In our work, it is aimed to propose a solution by taking into account that agents in the swarm has simple structures and limited sensor capabilities.

To achieve these objectives and meet these requirements, the goals for this thesis work

are defined in the following subsections of 1.3.

1.3.1 Heterogenous Robots with Different Dynamics

It is aimed to create a formation control system with heterogeneous agents to achieve complex tasks which require different functionalities. To accomplish this objective, it is assumed that agents have different dynamics from each other like different friction surfaces, geometrical structures, dynamics and functionalities. They have different volumes and masses and they may collide with the other ones and the obstacles in the environment.

1.3.2 Communication Infrastructure

One of the objectives in this work is to implement a localization solution for the agents. Also it is required to implement a communication backbone to execute a decentralized decision making process in which each agent reaches a global consensus. To meet these requirements, it is needed to implement a communication infrastructure in the proposed solution. By taking into account that the agents in the swarm have limited communication capabilities and can only negotiate with its local neighbors in a narrow line of sight range due to power consumption issues and their weak radio links, this infrastructure must have following specifications.

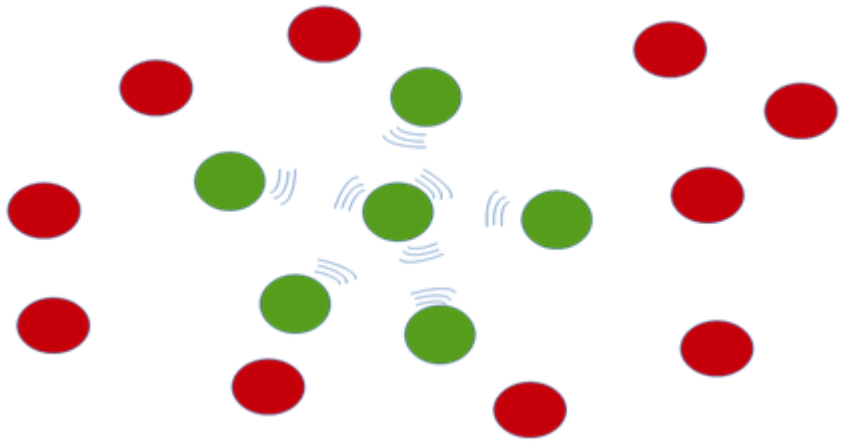


Figure 1.6: Radio Links on Agents Have a Narrow LOS Range

a)Communication Topology

Communication topology is a wireless mesh network in which each agent relays data for the rest of the network. The network is fully connected and has routing technique where the data is propagated along a route by transporting over the nodes(member agents of the swarm).

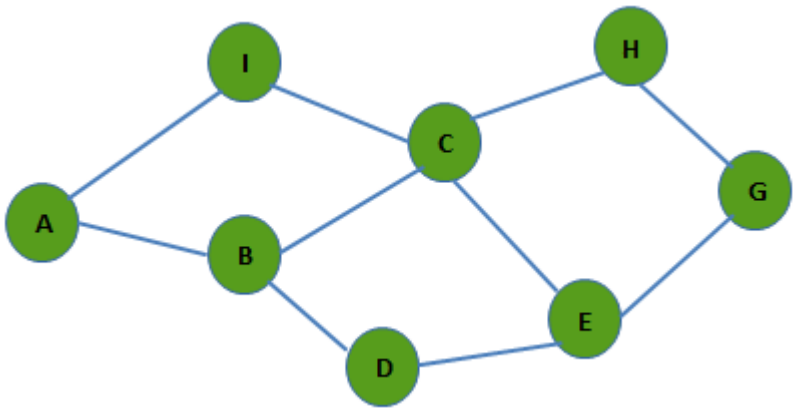


Figure 1.7: Mesh Network Between Agents

b)Communication Bandwith

Bandwidth of the communication between agents is limited and nodes can only transport most critical data like heartbeats, agent IDs, type and position etc.

1.3.3 Decentralized Decision Making Process

Centralized formation controller systems implement a single controller server/root node to process all the data needed to achieve the desired objectives. This type of systems achieve superior performance and optimal decisions but they require high computational power, high communication bandwidths and are not robust due to dependence on a single controller [11]. Decentralized formation controller structures have agents which are completely autonomous and responsible their own individual decisions. In this work, a hybrid centralized/decentralized controller architecture in which there is a central manager which partitions the desired formation shapes into goal states and there are independent agents who make their own choices on these goal states as unaware of the other agents' choices.

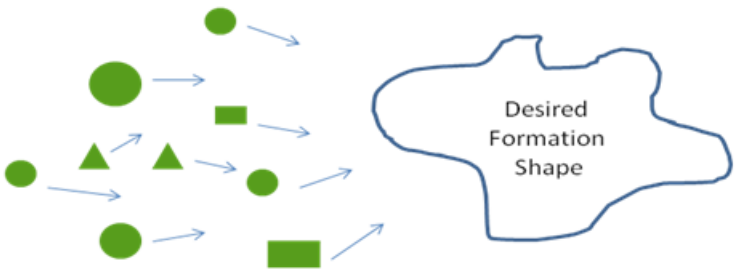


Figure 1.8: Agents make their own choices about target goal states

1.3.4 Complex and Dynamically Changing Formation Shapes

In literature, formation control systems are generally designed with a single or combination of simple geometrical shapes and they require analytical expressions of these shapes while implementing the proposed control laws for individual agents [20]. Specific tasks which requires more complex shapes cannot be achieved with this approach. In our work, we have focused on designing a formation control algorithm which does not require the mathematical definition of the desired formation shape. The goal to accomplish this objective is to design a solution which does not depend on the analytical expression of the formation shape. Furthermore, in literature the general approach is to implement a solution to cover a static formation shape. In real world applications it may be needed to update the desired shape dynamically according to the requirements of the task. In our work, one of the goals is to propose an algorithm which can adapt itself to dynamically changing formation shapes.

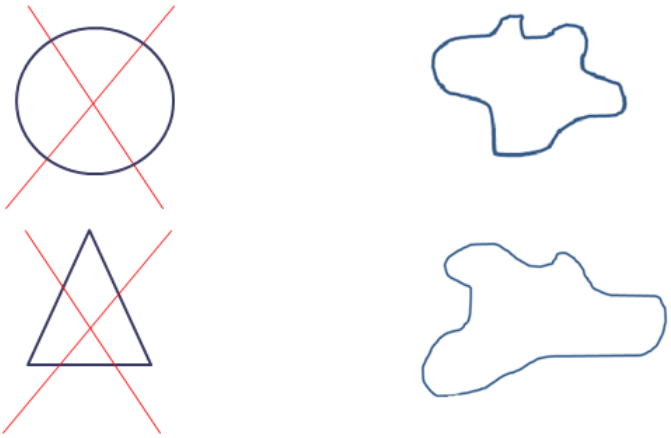


Figure 1.9: Complex and Dynamically Changing Formation Shapes

1.3.5 Simple Agents with Low Sensor Capabilities and Low Computing Powers

Agents in the swarm are assumed to have low sensor capabilities and weak computing power [4]. These constraints must be taken into account during the control system design, since individuals do not have a high resolution and sensitive data about their states, and they cannot execute high level complex control algorithms.

1.4 Requirements

The goals of this project define some requirements about the implementation of the formation control system. These requirements are listed as follows:

1. Agents have to propagate their positions and velocity states with the help of inertial measurements.This process is handled with a Kalman estimator which takes the translational acceleration data as an input to the observer model .The translational acceleration data is calculated with the help of AHRS system composed by 3 axis accelerometers, gyroscopes and magnetometers.

2. Agents have to update&adjust their position data with the help of agents which have positioning sensors(position beacons) by local trilaterations.This position data is used in the estimator system as external measurements to correct the drifts caused by propagation error of translational accelerations. The ordering for this trilateration process have to determined appropriately to minimize the error on calculated position data of agents which are far away from the position beacons.

3. Agents have to update their route tables to create a communication backbone with the mesh network topology. Since agents are assumed to have low range&bandwidth radio links, the propagation of a data between each agent in the swarm will be handled over this mesh network.

4. Agents have to determine the goal states in the desired complex formation shape to cover with the help of a central server. Desired formation shape will be partitioned into potential goal states assigned to different types of agents. Performance analysis on proposed shape partitioning methods have to be done with some different criterias.

5. Assignment of the agents to their target goal states should be handled to minimize the total displacement of the agents while travelling towards the desired formation shape.

6. Simulations should be performed to compare the efficiency of different methods proposed in this thesis work. Different types of agents have to be represented with different dynamical and physical models during simulations.

7. Hardware demonstrations should be performed to illustrate the applicability of the proposed solution in real time systems. These applications may not contain the full implementation of the proposed system, but they must demonstrate a proof of concept(POC) environment.

1.5 Methodology

During the first part of the project, a local positioning system(LPS) is designed. In this system, agents which does not have position sensors, propagate their position and velocity states with their inertial measurements. Due drift problems on this solution, a position update&adjust process is handled with the help of position beacons which are agents with position sensors on their boards in the swarm. During the update phase of the solution, route tables for individual agents are determined with the help of Graph Theory based on Destination-Sequenced Distance Vector Routing Protocol (DSDV) algorithms. This process provides the required information to compose the clusters around position beacons and provides rank information for the agents which are in same clusters. Position measurements are handled with local trilateration process in a turn with the rank values for every agent around each cluster after the establishment of route tables. A Kalman estimator system is designed to fuse these propagation and update phases of the solution.

On the second part of the thesis work, formation controller system is designed with two novel methods based on bubble packing method and randomized fractals method. Desired complex formation shapes are partitioned into goal states according to the heterogeneous agents in the swarm for both of these two methods. Decision process of the agents about their target goal states to optimize the total displacement of the

swarm is implemented with the help of Visibility Graphs and Hungarian algorithms. Internal velocity controllers for individual agents to reach the desired target goal states by providing obstacle avoidance, are implemented with a full state feedback method by regulating the augmented dynamical system with the gains optimized by Linear Quadratic Regulators (LQR).

1.6 Contribution of Thesis

The main contributions of thesis are:

1. Designing a local positioning system(LPS) based on local trilaterations to provide high accuracy position data to the agents which do not have specific position sensors on their boards.

2. Implementing a wireless mesh network between agents in the swarm and design a communication infrastructure and related routing algorithms to exchange the local data globally in the network

3. Implementing a formation control system with complex and dynamically changing formation shapes by using swarms composed of heterogeneous agents.

4. Designing and implementing the rules&algorithms for the decentralized decision making process of the individual agents about the goal states in desired formation shape.

1.7 Outline of the Thesis

This thesis work is organized into 6 main sections .Chapter 1 introduces the main theme and the potential usage areas of formation control, while specifying our motivation and the requirements&problems to meet&solve related with the topic.

Chapter 2 gives literature reviews about the related works and discuss about the advantages&disadvantages of the proposed method.

Chapter 3 introduces the methods and solutions used in two different parts of the

problem; local positioning system and formation control system. In this chapter, routing algorithms and mathematical aspects of the trilateration process is introduced. On the other hand methods&algorithms used for formation control is discussed in details.

Chapter 4 provides simulation analysis on the local positioning system and gives mutual evaluations of the performances of different methods used in formation control system.

Chapter 5 provides and discuss the details of hardware implementations and the experimental results.

Chapter 6 concludes the thesis and defines the future works related with the thesis.

CHAPTER 2

LITERATURE SURVEY

Formation control requires local positioning, detection of the general shape, task allocation to individual agents and task dispatching. The literature has related works in these different fields but lacks the adaptation to dynamically changing shapes. This chapter is intended to motivate the need for the topic of this thesis by underlying the gap, it has filled by reviewing existing works in related fields.

2.1 Formation Control Systems

Formation control problem have different subproblems like formation shape generation, formation reconfiguration&selection and formation tracking [11]. In formation shape generation, agents are expected to get a formation shape which can be defined by external commands or with some mathematical constraint functions [21]. One general approach is to consider artificial potential functions. Samitha and Pubudu [22] have presented an artificial potential function method based on the consideration of the problem as controlling the position of a swarm into a shape, bounded by a simple closed contour. This approach results in deploying uniformly of swarm agents within the contour. Their work provides analysis about the stability and robustness of the proposed system based on Lyapunov like functions. Figure 3.92 illustrates a simulation environment of a swarm getting the desired pentagonal like formation shape in presence of obstacles. Here the desired formation shapes is defined with an analytical expression and individual control laws for agents are composed with artificial potential functions using these analytical expressions. In real world applications, it

may not be possible to have the analytical expressions of the desired formation shape. In our thesis work, we have focused on designing control laws which do not depend on the analytical expressions of the formation shapes.

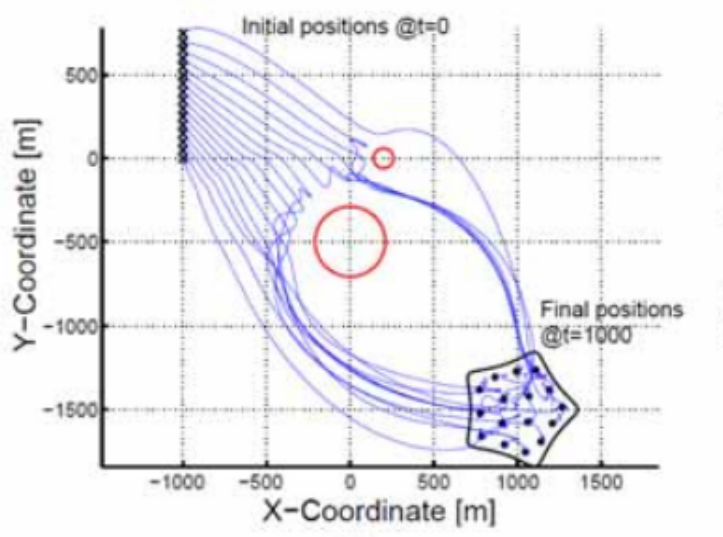


Figure 2.1: Motions and Formation of the Agents in Presence of Obstacles [22]

In some applications, it may be needed to change the formation shape or splitting and joining of the agents together due to either a change in coordinated task requirements or change in environmental conditions such as narrow corridors. In such a scenario, the swarm has to propagate in a narrow corridor before reaching the desired goal state and it is not possible to follow this path by keeping the final desired formation shape. Hence, swarm has to adapt itself to environmental conditions while following a predetermined trajectory. This task requires formation reconfiguration and dynamic task assignment of swarm agents to be dispatched. Hou et al. [7] have defined a method based on global objective functions to provide formation control of a swarm. In their approach it is possible to implement scaling and rotating functions into control laws to adapt the swarm to environmental conditions while achieving a specific task.

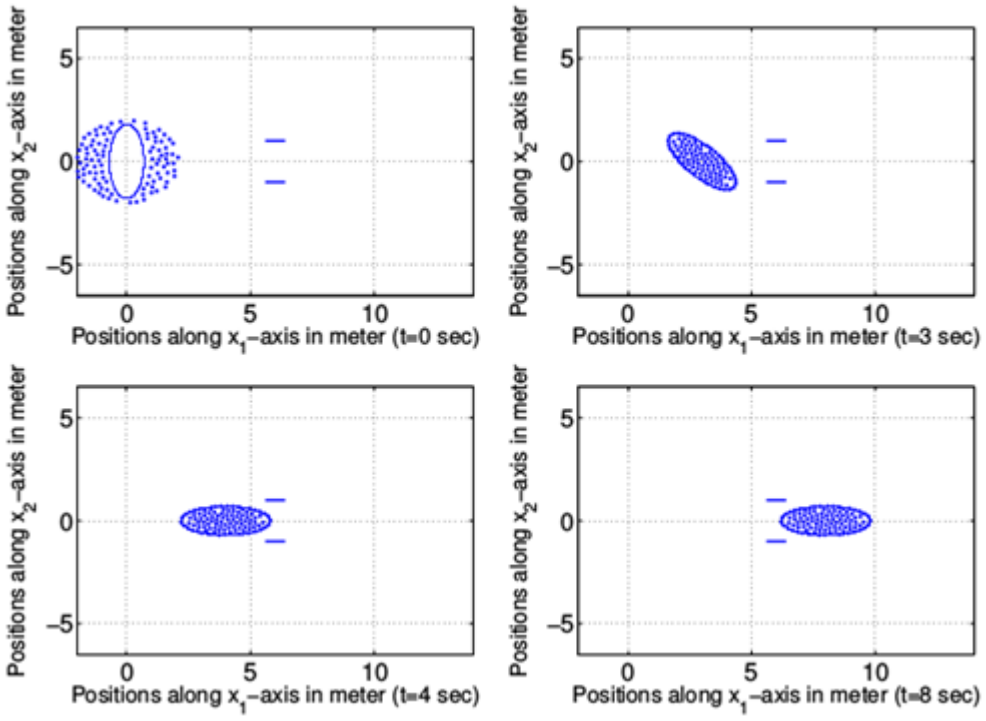


Figure 2.2: A Swarm in Rotating and Scaling Ellipse Formation [7]

Figure 2.2 illustrates a simulation environment with a swarm of 100 mobile agents with an ellipsoidal formation shape. The formation shape is rotated dynamically to follow a path through a narrow corridor. Swarm has to adapt this rotation maneuver to follow the desired trajectory. This kind of dynamical adaptations are always required in complex tasks in real time applications with complicated workspaces. Their approach also requires the analytical expressions for the desired formation shapes.

One of the subproblems studied in formation control is formation tracking. The main objective of this problem is to maintain a desired formation with a group of robots, while tracking or following a reference trajectory. The solutions for the formation tracking approaches can be classified into three basic strategies as leader-following, virtual structure and behaviour based approaches [11]. The most general strategy to provide a solution for this problem is leader-following swarm structures [23]. Other strategies have a basis on optimization and graph theory approaches [11]. Kumar et al. proposed a vision based formation control framework for this problem. This framework has a leader following infrastructure [23]. Figure 2.3 illustrates a simu-

lation environment of this leader follower approach on formation tracking problem. The individual agents achieve a coordinated turn by keeping their relative positions to their local neighbors.

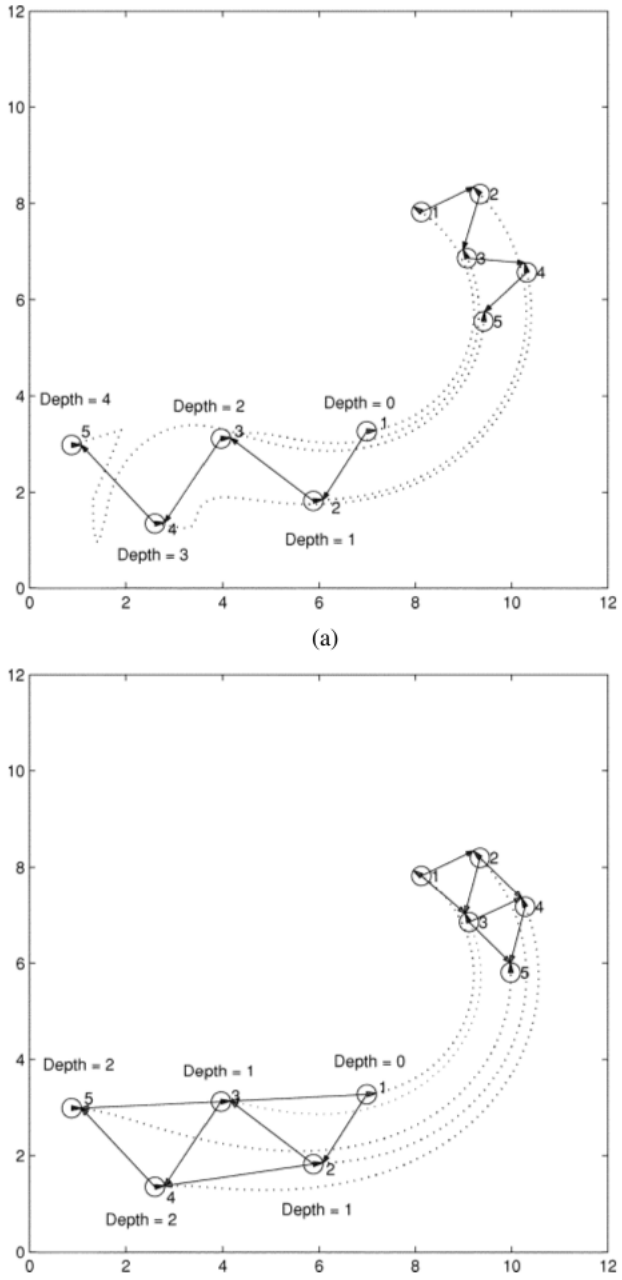


Figure 2.3: Five Robot Formation With Trajectory Tracking [23]

In leader following strategy, some of the agents in the swarm are the leaders to manage the rest of the swarm to achieve a desired specific task and the rest of the agents act as followers. This approach reduces the formation control problem into tracking control

problem of individuals to follow the leader from a desired distance and bearing angle, thus the stability and convergence analysis of the formation can be done with the usage of single tracking controllers of members. This approach simplifies the tracking problem of a network of agents to a single agent. Kumar et al. at [23] proposed a control framework in which follower agents move along a trajectory afterwords the leader agent with a desired separation distance l_{ij} and desired relative bearing angle ψ_{ij}. Figure 2.4 represents a formation control with three agents where R3 is the leader and R1,R2 are the follower agents.

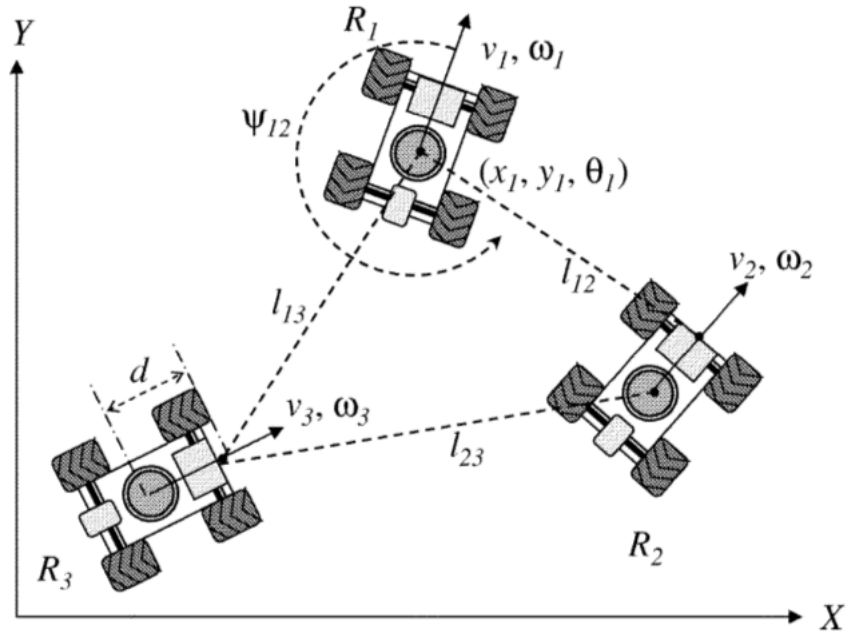


Figure 2.4: Leader-Follower Systems [23]

In this approach it is hard to gather the agents in a certain shape. Another drawback is that, determining the separation and bearing angles for individual agents is getting harder as the number of agents in the swarm increases and this strategy is not fault tolerant to the absence of communication between agents.

In virtual structure approach, the formation is composed with a virtual rigid body. Formation control is applied to the whole virtual structure and then the individual agent control laws are determined with inverse dynamic solutions [11]. Lewis and Tan [24] proposed a virtual structure based method for formation control with a bidirectional flow control where robots move to keep the virtual structure when the swarm

is following a trajectory and virtual structure adapt itself to the robots' current positions to compensate the relative errors at the end of a maneuver. Figure 2.5 illustrates such a maneuver implemented with their approach.

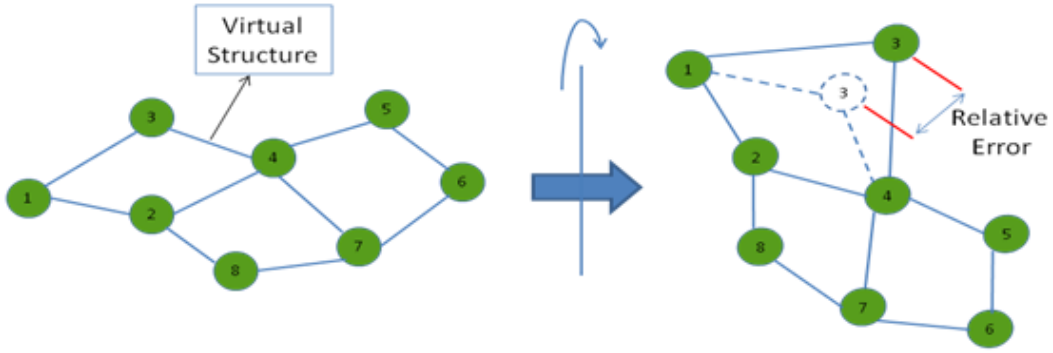


Figure 2.5: Maneuver of a Formation and Compensation of Virtual Structure

In virtual structure strategy it is easy to achieve a coordinated behaviour for the group to maintain the formation during a trajectory tracking or a maneuvering, but it is not a suitable strategy to apply a formation control to achieve certain geometrical shape with the swarm agents.

Behaviour based strategies model every agents' behaviours to achieve specific tasks with swarm. These behaviours may be very simple like randomly walk and avoidance of obstacles in the environment or they may be defined in a very complicated manner in order to achieve complex formation shapes with the entire swarm while for example optimizing the overall energy consumption depending upon the implementation of the controller structures. One of the main usage of this strategy is artificial potential field based implementations . Cheng and Nagpal have introduced a robust and self repairing formation control method for swarms [25]. In this approach, individual control laws for the agents are defined with the artificial forces between agents themselves (to avoid collisions) and between the desired formation shape and agents. This solution provides robustness to the agent losses in the swarm during formation control and the rest of the swarm has the ability to refiil their absence in real time without changing the dynamics and the parameters of the formation controller. Because individual control laws are not dependent on the other member of the swarm. Each agent can calculate its own control input at an instant time with current forma-

tion shape and current members of the swarm and the whole swarm converge another equilibrium with current members [25]. Figure 2.6 shows a simulation output of their work. Some of the agents from the right bottom corner of the desired formation shape are removed away from the swarm and the rest of the agents refill their absence during runtime. One of the main disadvantage of the artificial potential based approaches is that , the control forces applied to individual agents are determined instantaneously in accordance with that agent's and the other agents' positions and they cannot guarantee the optimization of the total distance travelled by the agents. Another drawback related with this type of solution is that, there is a possibility to have local minimas in the solution where an agent reaches an undesired point in configuration space under the equilibrium of different types of artificial force components. In that case the total control input acting on the agent will be zero because of cancelling force vectors which has opposite directions to each other generated by formation shape and obstacles etc. In this strategy, the solution may converge slowly to the steady state due to absence of generalized goal states for individual agents in the final formation shape. Because, there are no specific goal states for the individual agents to reach at the final configuration and they are expected to get a global equilibrium with the help of different artificial force components.

Figure 2.6: Formation Control with Artificial Forces [25]

Another approach is to define mathematical constraints and objective functions to achieve a specific formation shape by controlling the shape of the swarm colony while following a trajectory. Kumar and Belta [26] presented an abstraction method of configuration space to a manifold defined as $A = GxS$ where G is a lie group representing the position and the orientation of the swarm and S represents the shape of the manifold. They provide individual control laws which can be separately handled to ma-

nipulate the lie group G to achieve formation tracking and orientation control and to manipulate the shape S to achieve different geometrical shapes.Their method define the desired formation shape with shape manifold equations and control the orientation and the scale of this shape with lie group. Figure 2.7 shows a simulation output in which the shape is defined as a rectangular area. The orientation and the scaling of the formation shape is dynamically changed to adapt the swarm different environmental conditions. Similarly Cheah and Slotine [7] proposed a method based on objective functions . Common drawback for these researches, they can only implement a limited number of simple geometrical shapes because the desired formation shapes must be analytically identified in order to compose the related objective functions or shape manifolds. Even if it is possible to define a simple geometrical shape and to control the rotation and the scaling of this shape dynamically, there may be need for more complex and dynamically changing formation shapes rather than scaling and rotation maneuvers in real time applications.

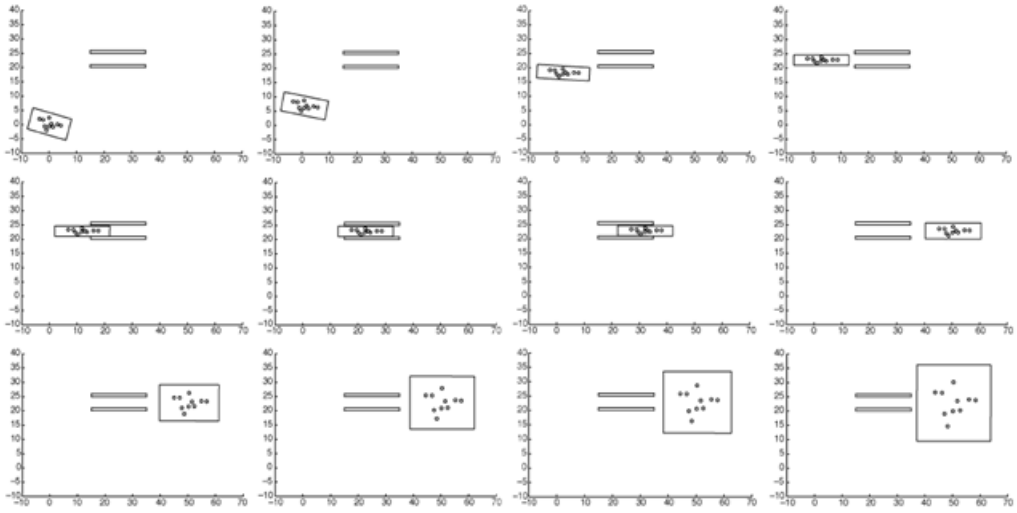


Figure 2.7: Formation Control with Objective Functions [26]

2.2 Partitioning Complex Geometrical Shapes

In this thesis work, one of the proposed solutions for the formation control problem is to partition the desired complex shapes into subsets of goal states for each type of agent in the swarm. Main idea is to dynamically assign the agents to these goal states

on runtime to minimize the overall displacement of the agents. Shape partitioning process is used to determine the goal states of the agents within the formation to cover the desired complex geometrical shape. There are some different solutions in the literature including fractal filling of space algorithms and bubble&circle packing algorithms.

Fractals are self similar patterns in all scales of themselves. They are defined with simple rules and they can cover any complex shape in the nature by progressing this simple rules iteratively. This approach is widely used in mesh generating algorithms and space filling problems. Shier and Bourke [27] have introduced a randomized fractal filling of space algorithm. They proposed a fractal based method to cover a given geometrical shape with the desired shapes and they provide the proof of their algorithm is space-filling with the following equations. Let A be the total area to be filled. In our case, A will be the area of the desired formation shape. If we choose the area of the fractals sequentially (i.e. bubbles which are representing the coverage area of the agents) while filling the desired formation shape as A_i with the following equation

$$A_i = \frac{A}{\zeta(c,N)(i+N)^c} \tag{2.1}$$

where $\zeta(c,N)$ is the Hurwitz zeta function defined by

$$\zeta(c,N) = \sum_{i=0}^{\infty} \left(\frac{1}{(i+N)^c} \right) \tag{2.2}$$

This Hurwitz zeta function will converge to zero while $i \to \infty$ with selection of $c > 1$ and $N > 0$. In view of Equation 2.1 one can write by replacing the Hurwitz zeta function with the definition given in Equation 2.2

$$\sum_{i=0}^{\infty} A_i = \sum_{i=0}^{\infty} \left(\frac{A}{\zeta(c,N)(i+N)^c} \right) = A \tag{2.3}$$

such that the sum of all areas A_i is the total area A to be filled, that is, if the algorithm does not halt then it is space-filling. Thus, it is possible to fill the desired shape with fractals chosen with areas defined in Equation 2.3. Some of the outputs of their algorithm is given at Figure 2.8. The desired shapes are filled with rectangular and star shaped fractals by selecting the areas as described in Equation 2.3.

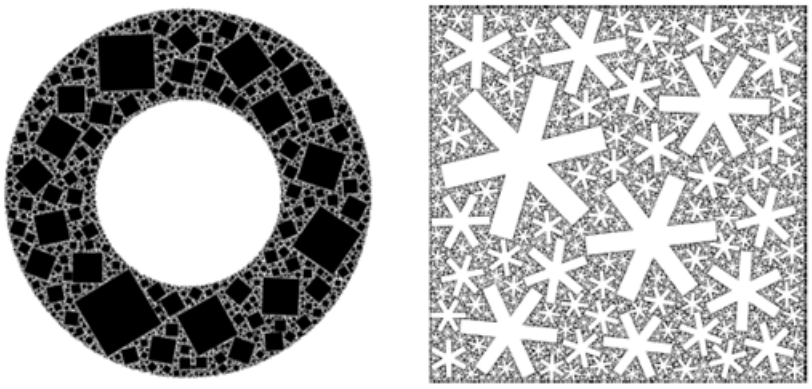


Figure 2.8: Space Filling Examples with Randomized Fractals [27]

Bubble&Circle packing algorithms are widely used for mesh generation problems in finite element method. Shimada and Gossard [28] proposed a method based on interbubble forces to provide close packaging of bubbles in desired geometrical shape. They define artificial interbubble forces to avoid collisions between bubbles and shape forces to keep the bubbles at the interior region of the desired shape. This idea is similar to the potential field based approaches in formation control problem since they both define artificial force components on individuals to cover a desired shape homogenously. The related interbubble forces are described at Figure 2.9. These forces have a nonlinear function of distance between the center of each bubbles and if this distance is larger than l_o, the interbubble forces are negative which means they attract each bubble towards the other ones. Interbubble forces increase while the distance between bubbles are decreasing to provide collision avoidance between bubbles. l_o is the stable distance in which there is no interbubble force acting on bubble and it reaches an equilibrium point in the workspace.

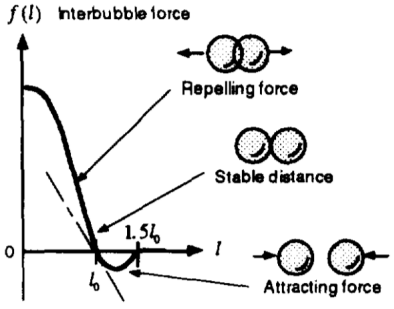


Figure 2.9: Interbubble Forces [28]

The main idea is provide a close packing of bubbles which mimics a pattern of Voroni tessellation, corresponding to well shaped Delaunay triangles or tetrahedras.Figure 2.10 shows the corresponding voronoi polygons and delaunay triangles of a set of packed bubbles. The center of the cells in voronoi polygons are selected as the centers of the bubbles. Delaunay triangles corresponding to the set of uniformly packed of bubbles are well-shaped triangles [28].

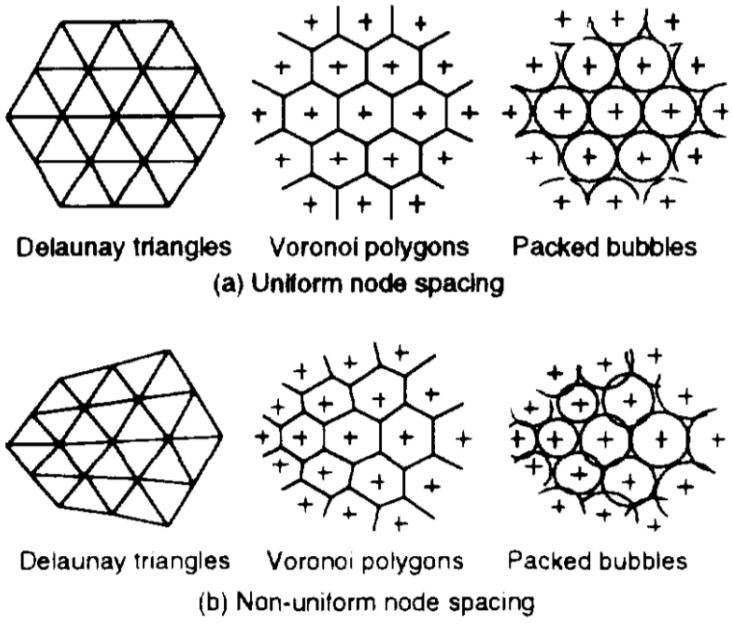


Figure 2.10: Uniform and Non-Uniform Node Spacing [28]

With the help of these interbubble forces, they provide an adaptive bubble packing algorithm for mesh generation. A result with a 2D shape is given at Figure 2.11. At the end of 50 iterations, bubbles cover the desired shape with uniform spacing with the help of interbubble forces and the resultant delaunay triangulation corresponding to the set of packed bubbles are well-shaped triangles.

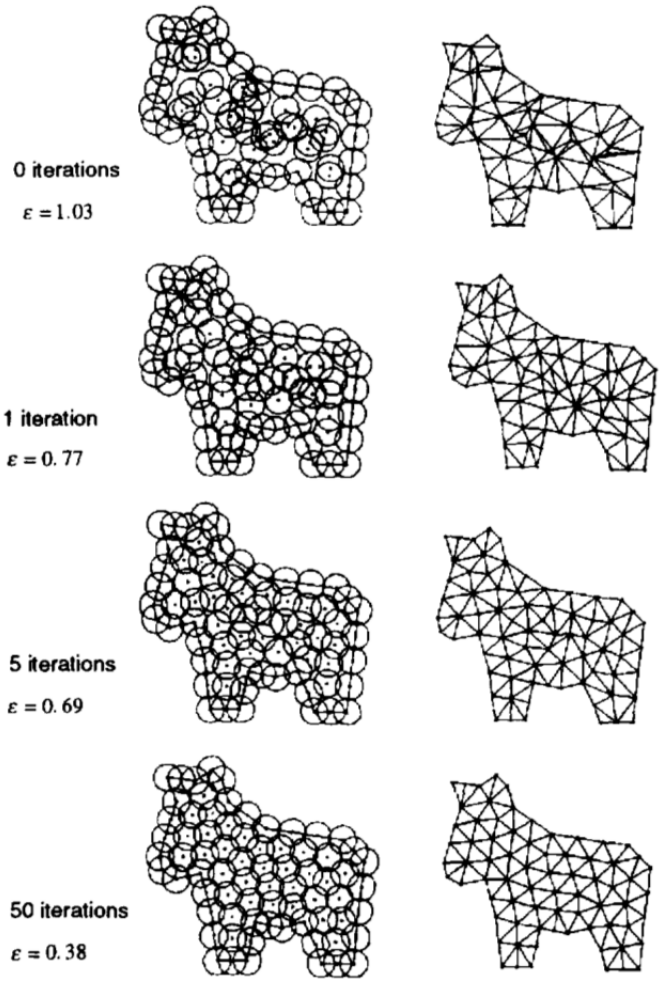


Figure 2.11: Mesh Generation with Interbubble Forces [28]

This approach can easily be augmented for different types and number of shapes to partition a complex geometrical shape with regular sets. It is possible to generate a mesh with different shapes which represents different types of agents in the swarm, rather than using bubbles with same radius. With this adaptation on the proposed method, it is possible to partition the desired shape into goal states which are subsets of different types of agents in the swarm with the help of interbubble forces.

2.3 Local Positioning Systems

Positioning systems are used to provide the required position data to the systems where it is desired to control the location of the mobile agent in the workspace. These

systems fall into two main branches, global positioning systems and local positioning systems. Global positioning systems(GPS) has become increasingly popular for a couple of last decades for tracking location. In outdoor formation control applications this solution is used for positioning [29]. It is a precise system depends on satellite based positioning mainly developed for direction finding and navigation. Some of the problems encountered with the usage of GPS systems: (1) the signal from the satellites cannot penetrate to the indoor space so it doesn't perform in such areas, (2) it looses its precision in rich scattering environments such as urban areas [10]. A local positioning system can provide a position information where GPS systems are unavailable,with the usage of signaling beacons which are placed at the exactly known locations. In indoor formation control applications this kind of approach is used for positioning [17].

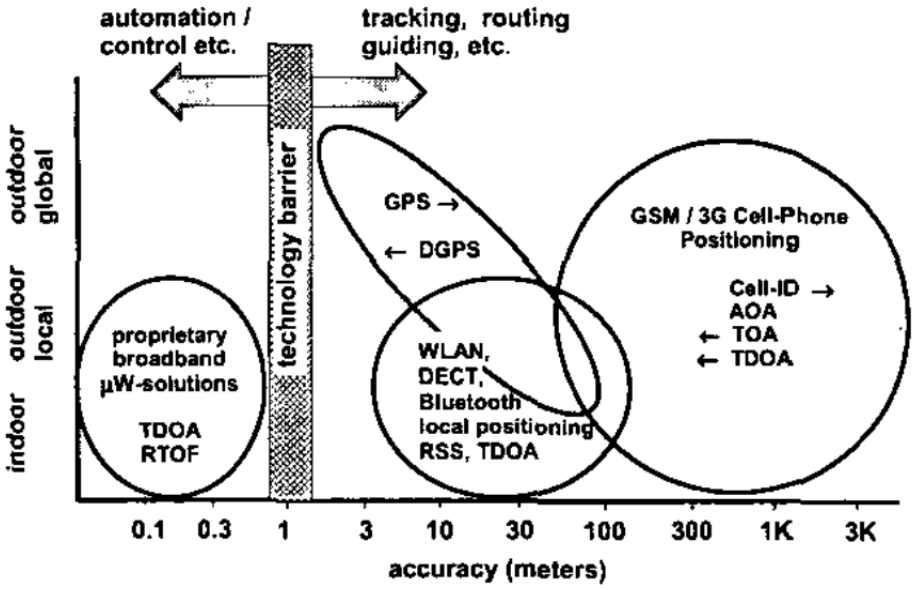


Figure 2.12: Accuracy Statistics of Different Positioning Sources [30]

Figure 2.12 represents an overview of current positioning systems [30]. Global positioning systems are widely used and they provide accuracies in the range of 3-30 meters. They further can operate in outdoor environments requiring radio signals from satellites. Differential GPS systems decrease these accuracy range below 3 meters with the help of additional local static beacons. GSM based solutions have the worst accuracy performance because they localize the nodes in the network by checking the relative signal strengths from different beacons, but they can perform in indoor en-

vironments partially [30]. Local positioning systems have the capability of working indoor environments because they use the signaling beacons which are placed at the exactly known locations [30].

Local positioning systems has different system topologies illustrated in the Table 2.1.

Table 2.1: Local Posititioning Systems with Different System Topologies [30]

Concept	Concept Definition
Remote Positioning	Measurement from remote site to mobile device
Self Positioning	Measurement from mobile unit to usually fixed transponders(landmarks)
Indirect remote positioning	Self positioning system with data transfer of measuring result to remote site
Indirect self positioning	Remote positioning system with data transfer of measuring result to mobile unit

Two main topologies are self positioning and remote positioning systems [30]. In a self positioning system, a mobile device finds its own position with the help of a predetermined reference such as a starting point or a beacon node with exactly known positions. On the other hand, in remote positioning systems a mobile node locates other objects' positions relative to its own position [10]. These two type of topologies can be converted to each other with the help of communications structures integrated on the devices to share the result of position measurement and thus indirect remote positioning and indirect self positioning system topologies can be implemented. Self positioning systems can be implemented to provide position data to the individual agents in formation control problems. The approach is to distribute the position data to the individual agents from a beacon node with exactly known positions. The individual agents which do not have a positioning sensor on their boards, measure their distance to these beacons and correct their position data with these external measurements. Beacons are assumed to have a GPS sensor on their boards for outdoor applications. They are assumed to have a special tag to locate them in an indoor

environment with the help of external devices such as an E/O camera.

2.3.1 Measurement Principles

Local positioning systems can be used to provide position data in formation control problem by using robot networks. These systems differ mainly based on their measurement approaches which are the angle of arrival (AOA), received signal strength(RSS) and propagation-time based systems used in local positioning systems.

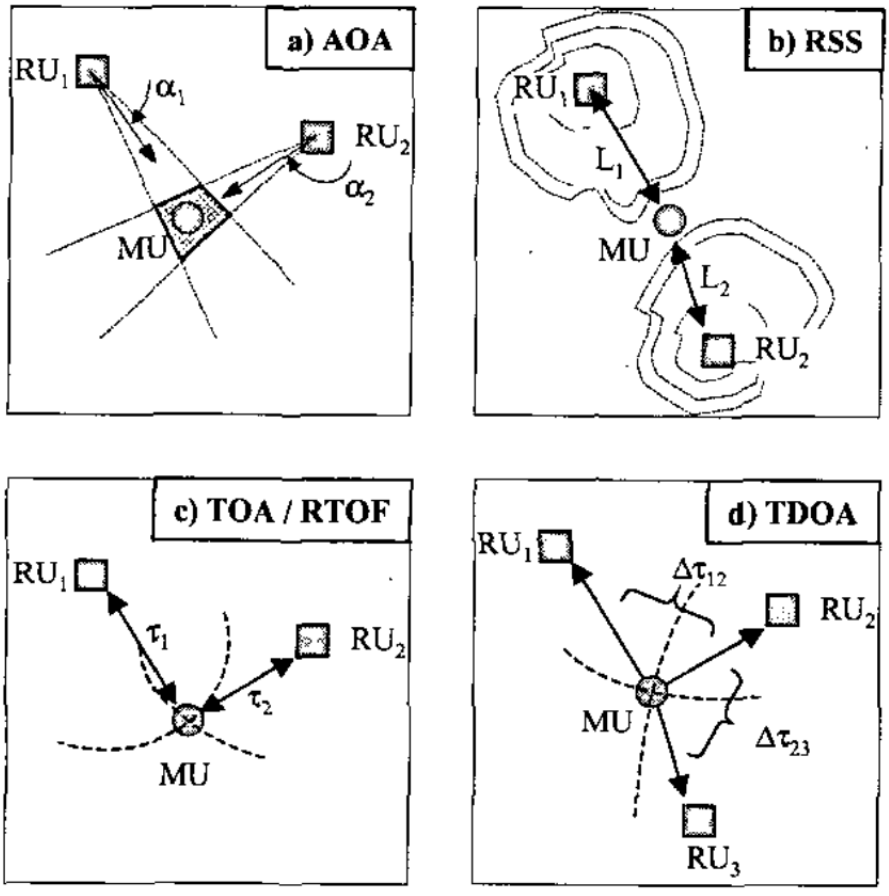


Figure 2.13: Different Measurement Principles [30]

Angle-of-arrival (AOA) systems use directional antennas to measure the bearing angle to the points located at known positions are measured. The position value of device can be calculated with the intersection of several measurements, but the accuracy is limited by shadowing and multipath reflections of radio signals [30].

Received signal strength (RSS) systems calculate the distance value by taking the difference of the received signal power from the transmitted power. Some advanced propagation models are required to calculate the distance from the transmission loss in the air to eliminate the multipath fading and shadowing effects [31].

Time based systems calculates the distance between measuring unit and signal transmitter with the help of propagation time like used in the global positioning systems generally. This process requires a perfect time synchronization between the mobile and stationary units [30].

In this thesis work, we implement a self positioning system in which every agent localize itself with the help of position beacons in the swarm with exactly known positions. The distance from the agents to these beacons in the swarm are assumed to be calculated with the help of a time of arrival(TOA) solution in which a node can calculate its distance to the transmitter beacon by measuring the difference between the timestamps of transmission and reception of the signal.

2.3.2 Trilateration Process

Trilateration process is used to determine the position of unknown locations with the help of distance measurement to known positions [32]. In self positioning systems, the main approach is to calculate the position of mobile agents with the help of distance measurements to the beacons with exactly known positions. This position calculations are handled with trilateration process. It is widely used in wireless sensor network topologies and local positioning systems. In theory, it is needed to have at least four beacon nodes to calculate an unknown position in 3D, and at least three beacon nodes to calculate an unknown position in 2D environment. But these worst case numbers are generally not sufficient to estimate an unknown position with a good accuracy due to errors on range calculations and synchronization problems. Figure 2.14 demonstrates a simple trilateration process in 2D environment with the help of three position beacons. Suppose a mobile device which tries to estimate its position with the help of local positioning system is at the red point in the figure. If it can measure its distance to the beacons named A,B and C with exactly known positions, it will be possible to estimate the unknown position of this mobile device

with the same approach used in global positioning systems.

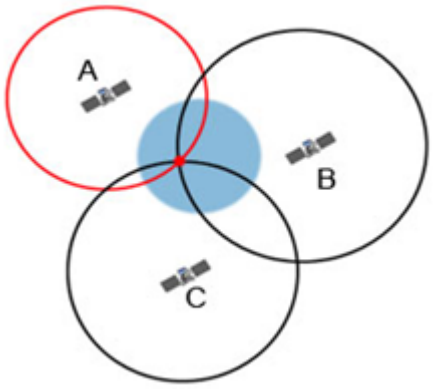


Figure 2.14: Trilateration Process [33]

CHAPTER 3

METHODOLOGY

In this thesis work, the problem of dynamical formation control of heterogeneous mobile robots is reduced down to two subproblems such as local positioning and formation control. In our implementation there are two main types of agents in the swarm. First type agents have position measurement sensors on their boards. We call these agents position beacons. Second type agents do not have position sensors on their boards. Local positioning subsystem proposes a solution for the localization of agents in the workspace for second type of agents. On the other hand, formation control subsystem implements algorithms to cover the desired formation shape homogeneously with the agents in the swarm. In this chapter, our solutions for these subproblems are presented in details.

A general block diagram of the system design is illustrated in Figure 3.1. According to this diagram, local positioning system provides a basis for the formation control problem with agents' state vectors composed of translational positions and velocities. Basically, this subsystem provides position and velocity data to the second type of agents in the swarm. Formation control subsystem proposes a solution to achieve desired complex shapes with heterogeneous agents in the swarm. Formation control system gets the requested formation shape externally and creates individual control laws for agents to cover the desired shape homogenously. The implementation details of these subsystems are given in the following sections.

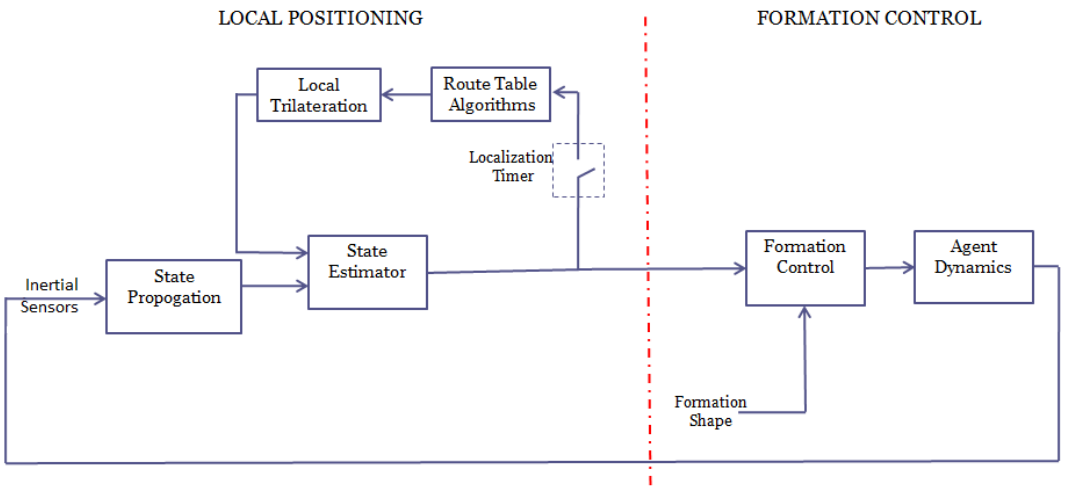


Figure 3.1: General Block Diagram of the Provided Solution

3.1 LOCAL POSITIONING SYSTEMS

Local positioning system is a subsystem to provide a complete solution to the localization of the second type agents in the workspace. As discussed in Section 1.3, agents are expected to have low sensor capabilities and only a limited number of them have external position measurement sensors on their boards and we call these agents position beacons in our project. The rest of the swarm which is composed of second type agents, has to maintain their position&velocity data with the help of their inertial measurements and local trilateration process. Propagation of the states with these inertial measurements are always inclined to drift problems due to the error&noise and bias problems of the sensors [34]. Because of this problem, we have corrected these state vectors with an external measurement provided by local trilateration process. We have proposed a complete solution for this subsystem as illustrated in Figure 3.2.

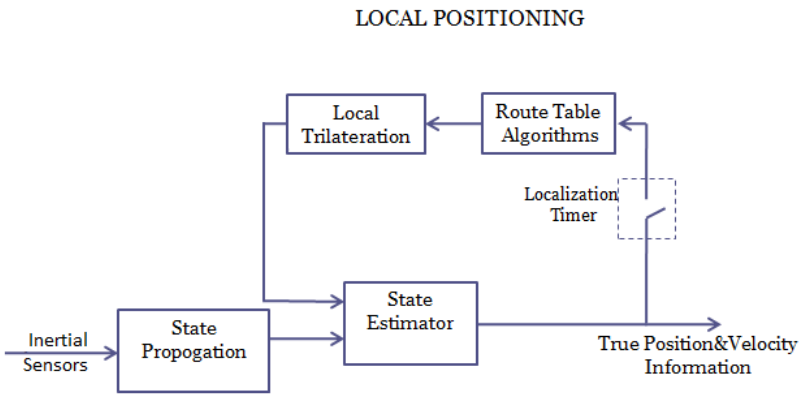


Figure 3.2: Local Positioning System

As illustrated in Figure 3.2, agents propagate their states with the help of inertial measurements. This state propagation process is executed with high speed data provided by the inertial sensors. On the other hand, trilateration and route table determination processes require iterative and time consuming algorithms and their implementation details are given in the following sections. It is not possible to execute the trilateration and route table determination algorithms with the same execution frequency of state propagation process. Because of this reason, a localization timer to provide the minimum required time for the route table determination and trilateration process is implemented to the solution. Agents will correct their state vectors with the localization period by measuring their positions with trilateration process and execute the update procedure in their state estimator algorithms. The requirement about the maximum value of the localization period is determined with the help of Monte Carlo simulations discussed in Section 4.2 by defining a maximum allowable error on the position data.

Local trilateration process requires distance measurements to the position beacons which are direct neighbors of an agent. Route table algorithms are used to provide information to the agents about the position beacons whether they are direct neighbors or not. On the other hand, route tables also implement a mesh network between agents. This network is used to transport data globally over the network. It is composed with a bidirectional graph in which each agent can communicate with its direct neighbors. We have assigned an edge between two nodes if there is a radio link between them. This line of sight(LOS) radio link is the main criteria for the agents to

become direct neighbors in the network. Thus, having a physical Euclidian distance to an agent within the communication range is not enough to be the direct neighbor with that agent, i.e. even if two agents are very close to each other in the workspace, they are not direct neighbors if there is no radio link between them because of an obstacle preventing the line of sight. Figure 3.3 shows an example of a mesh network created with 8 agents in the environment.

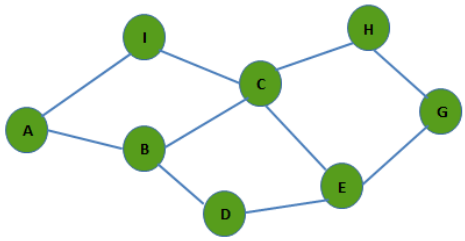


Figure 3.3: A Sample Mesh Network Created with 8 Agents

3.1.1 Local Trilateration Process

Trilateration process helps the second type agents to localize themselves with the help of position beacons which are their direct neighbors. Trilateration calculations use distance measurements to the position beacons with known positions, to determine the coordinates of second type agents [32]. These measurements are assumed to be done with the help of time of arrival(TOA) methods discussed in Section 2.3.1. In a TOA solution, an agent is able to measure the distance from itself to a position beacon only if it is a direct neighbor of that position beacon. Figure 3.4 illustrates simulation environment of a swarm which consist of 5 position beacons. Here agent C tries to localize itself with the help of distance measurements r_i and the position data B_i of the position beacons. It has 4 position beacons as direct neighbors in the network and this information is provided by the route table of agent C as defined in Section 3.1.2.

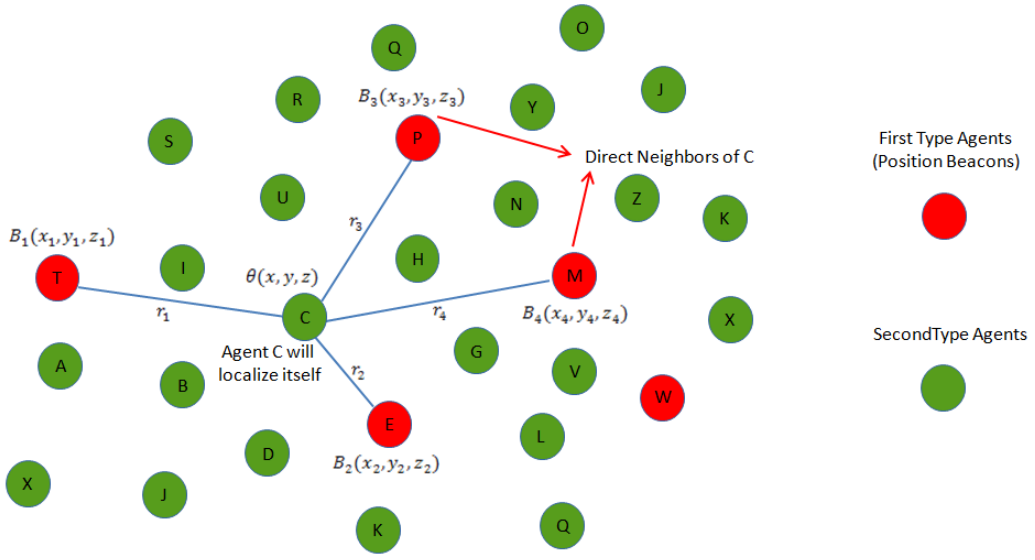


Figure 3.4: Environment for Trilateration Process

For the simulation environment illustrated in Figure 3.4, the distance function to any position beacon can be written as follows [32];

$$r_i = \sqrt{(x-x_i)^2+(y-y_i)^2+(z-z_i)^2} \quad (i=1,2,...,n) \tag{3.1}$$

where i denotes the beacon number and n is the total number of position beacons which is equal to 4 in our case. In general, we have n number of constraints in the solution of the localization problem. These constraints are defined as spherical functions for a localization problem in 3 dimensional workspace [32]. In our work, we have implemented a two dimensional localization solution with the assumption of each agent of the swarm conserve the same vertical position within the Earth centered coordinate system. With this assumption, the problem for the localization process has constraints with circle functions rather than spherical ones, presented with

$$(x-x_i)^2+(y-y_i)^2 = r_i^2 \quad (i=1,2,...,n) \tag{3.2}$$

It is possible to use these n circular constraints to calculate the unknown position of the agent C. For this purpose, lets assume $\theta = (x,y)$ is representing the coordinates of

agent C and $B_1 = (x_1, y_1); B_2 = (x_2, y_2); B_3 = (x_3, y_3); ...; B_i = (x_i, y_i)$ are the measured positions of the position beacons.

If any of these beacons is considered as the reference beacon and named with an index of r, the distance equations can be provided as following:

The distance between the agent C and any beacon i represented as

$$d_i(\theta) = \sqrt{((x - x_i)^2 + (y - y_i)^2)} \tag{3.3}$$

The distance between the reference beacon which is located at $B_r = (x_r, y_r)$ and the other beacons is evaluated as

$$d_{ir}(\theta) = \sqrt{((x_i - x_r)^2 + (y_i - y_r)^2)} \tag{3.4}$$

The distance between the agent C and the reference beacon is obtained as

$$d_r(\theta) = \sqrt{((x - x_r)^2 + (y - y_r)^2)} \tag{3.5}$$

Adding and subtracting x_r, y_r and z_r in equation 3.3 gives

$$\begin{aligned} d_i^2(\theta) =& (x - x_r + x_r - x_i)^2 + (y - y_r + y_r - y_i)^2 \\ =& (x - x_r)^2 + 2(x_r - x_i)(x - x_r) + (x_r - x_i)^2 \\ &+ (y - y_r)^2 + 2(y_r - y_i)(y - y_r) + (y_r - y_i)^2 \end{aligned} \tag{3.6}$$

Replacing the expressions in equation 3.6 by d_{ir} and d_r, the equation can be written as:

$$2((x_i - x_r)(x - x_r) + (y_i - y_r)(y - y_r)) = d_r^2(\theta) + d_{ir}^2 - d_i^2(\theta) \tag{3.7}$$

this general expression is valid for each beacon with i changing from 2 to n (by assuming that we have chosen the first position beacon with "$r = 1$" as reference beacon)

$$
\begin{aligned}
(x_2-x_1)(x-x_1)+(y_2-y_1)(y-y_1) &= \frac{1}{2}[d_1^2(\theta)+d_{21}^2-d_2^2(\theta)] \\
(x_3-x_1)(x-x_1)+(y_3-y_1)(y-y_1) &= \frac{1}{2}[d_1^2(\theta)+d_{31}^2-d_3^2(\theta)] \\
&\ldots \\
(x_n-x_1)(x-x_1)+(y_n-y_1)(y-y_1) &= \frac{1}{2}[d_1^2(\theta)+d_{n1}^2-d_n^2(\theta)]
\end{aligned}
\tag{3.8}
$$

if b_{ir} is defined for each beacon as follows:

$$
b_{ir} := \frac{1}{2}[d_r^2(\theta)+d_{ir}^2-d_i^2(\theta)] \tag{3.9}
$$

then the linearized system equations can be represented with $A\vec{x}=\vec{b}$ type equation where;

$$
A = \begin{bmatrix} x_2-x_1 & y_2-y_1 \\ x_3-x_1 & y_3-y_1 \\ \ldots & \ldots \\ x_n-x_1 & y_n-y_1 \end{bmatrix} \tag{3.10}
$$

$$
x = \begin{bmatrix} x-x_1 \\ y-y_1 \end{bmatrix} \tag{3.11}
$$

$$
b = \begin{bmatrix} b_{21} \\ b_{31} \\ \ldots \\ b_{n1} \end{bmatrix} \tag{3.12}
$$

with A is a $(n-1)$ x 2 matrix and b is a $(n-1)$ x 1 vector.

There are some possible solutions to this type of equation regarding with the structure of matrix A and vector b.

Solution to $A\vec{x}=\vec{b}$ Problem

In a localization problem handled in two dimensional space, the A matrix has $(n-1)$

rows and 2 columns, where n is the number of position beacons which are direct neighbors of an agent. There is no solution when the number of neighbors lower than 3 (i.e. $n < 3$) [32]. When the number of neighbor position beacons are equal or greater than 3 we have three different solutions according to the structure of the linearized equations.

1) Unique Solution:
If A is a 2 x 2 matrix (i.e. $n = 3$) and the its rank is 2, then the solution of $\vec{x}$ is unique with [35]

$$\vec{x} = A^{-1}\vec{b} \tag{3.13}$$

where $\vec{x}$ is the unique solution.

2) Minimum Norm Solution With Pseudo Inverse:
If A is a $(n-1)$ x 2 dimensional matrix where $n > 3$,which means the number of neighboring position beacons greater than 3, and if columns of A matrix form a linearly independent set (full column rank matrix) then the solution can be found with the projection of $\vec{b}$ over range space of A, $Proj_{R(A)}\vec{b}$ where [35]

$$Proj_{R(A)}\vec{b} = A(A^TA)^{-1}A^T\vec{b} \tag{3.14}$$

$$\begin{aligned} A\hat{x} &= Proj_{R(A)}\vec{b} \\ A\hat{x} &= A(A^TA)^{-1}A^T\vec{b} \end{aligned} \tag{3.15}$$

We can justify this expression by passing the righthand side term to the left

$$A(\hat{x} - (A^TA)^{-1}A^T\vec{b}) = 0 \tag{3.16}$$

then

$$\hat{x} = (A^T A)^{-1} A^T \vec{b} \tag{3.17}$$

$(A^T A)$ is invertible since A matrix is full column rank matrix, so

$$\mathscr{N}(\mathbf{A}) = \{0\} \quad and \quad \mathscr{N}(\mathbf{A})^{\perp} = \mathbb{R}^n \tag{3.18}$$

then

$$Proj_{\mathscr{N}(\mathbf{A})^{\perp}} \hat{x} = \hat{x} \tag{3.19}$$

this concludes that $\hat{x}$ is the unique minimum norm solution to the $A\vec{x} = \vec{b}$ problem

3) Minimum Norm Solution With Newton Iteration Method

If matrix A has the dimensions of $2\,x\,2$ or $(n-1)\,x\,2$ with $n > 3$ and if rank of A matrix is equal to 1 then the solution to the $A\hat{x} = \vec{b}$ problem can be found iteratively with the help of nonlinear least squares method that minimizes the cost function which is defined as sum of squares of the distance errors [32] :

$$F(\theta) = \sum_{i=1}^{n} \left(f_i^2(x,y) \right) \tag{3.20}$$

with

$$f_i(x,y) = \sqrt{(x-x_i)^2 + (y-y_i)^2} - r_i = f_i(\theta) \tag{3.21}$$

where r_i is the measured distance to a position beacon i and (x_i, y_i) is calculated position data of agent C iteratively.

There are various algorithms to minimize the cost functions in the literature. We have used Newton iteration to find the optimal solution in this thesis work. Because convergence of this method is quadratic in general and the performance of the algorithm

is extremely increased by choosing proper initial conditions [36]. We have used the propagated position data as an initial condition for the optimization process to make the algorithm terminated very fast with correct results. Algorithm is implemented as follows:

Considering $f(\theta)$ as

$$f(\theta) = \begin{bmatrix} f_1(\theta) \\ f_2(\theta) \\ \dots \\ f_n(\theta) \end{bmatrix} \tag{3.22}$$

We have defined the Jacobien matrix as follows:

$$J(\theta) = \begin{bmatrix} \frac{\partial f_1(\theta)}{\partial x} & \frac{\partial f_1(\theta)}{\partial y} \\ \frac{\partial f_2(\theta)}{\partial x} & \frac{\partial f_2(\theta)}{\partial y} \\ \dots & \dots \\ \frac{\partial f_n(\theta)}{\partial x} & \frac{\partial f_n(\theta)}{\partial y} \end{bmatrix} \tag{3.23}$$

Suppose we are trying to find the position of an agent defined with the vector of $\vec{R}$

$$\vec{R} = \begin{pmatrix} x \\ y \end{pmatrix} \tag{3.24}$$

To optimize the cost function, Newton iteration is implemented as follows;

$$\vec{R}_{\{k+1\}} = \vec{R}_{\{k\}} - (J_{\{k\}}^T J_{\{k\}})^{-1} J_{\{k\}}^T \vec{f}_{\{k\}} \tag{3.25}$$

where $\vec{R}_{\{k\}}$, $J_{\{k\}}$ and $\vec{f}_{\{k\}}$ denotes the variables calculated at k^{th} iteration. The explicit form of the equations can be derived by implementing our constraint functions to the generic statements, as follows;

$$J^T J = \begin{pmatrix} \sum_{i=1}^n \frac{(x-x_i)^2}{(f_i+r_i)^2} & \sum_{i=1}^n \frac{(x-x_i)(y-y_i)}{(f_i+r_i)^2} \\ \sum_{i=1}^n \frac{(x-x_i)(y-y_i)}{(f_i+r_i)^2} & \sum_{i=1}^n \frac{(y-y_i)^2}{(f_i+r_i)^2} \end{pmatrix} \tag{3.26}$$

and

$$J^T \vec{f} = \begin{pmatrix} \sum_{i=1}^{n} \frac{(x-x_i)f_i}{(f_i+r_i)} \\ \sum_{i=1}^{n} \frac{(y-y_i)f_i}{(f_i+r_i)} \end{pmatrix} \tag{3.27}$$

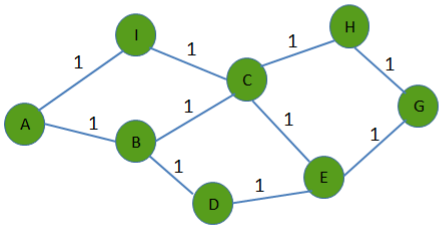
3.1.2 Route Table Determination

We have implemented a localization solution in which each second type agent is getting in trilateration process with the position beacons in the swarm as we have introduced previously. It is needed to be direct neighbors of the position beacons to get in trilateration process. In our project, route tables are used to provide information to the agents about the position beacons whether they are direct neighbors or not. On the other hand, they are used to create a mesh network in the swarm to provide a communication backbone. With this mesh network it is possible to send any data globally over the network to any destination by using agents as relays to transmit the data. In our implementation, each agent had its own route table in the network. These route tables include the information about the next nodes and costs to any destination in the network. Next nodes are representing the node which the data will be transferred next for that destination and costs are representing the minimum number of hops (i.e. number of relay agents on the path) for that destination. We have used the cost information for the position beacon destinations to check whether the agent is direct neighbor of that position beacon or not. In other words, the cost to a position beacon is equal to 1 in the route table, if that agent is the direct neighbor of the position beacon

3.1.2.1 Routing Algorithm- Bellman Ford

As mentioned in the Section 1.3, agents are assumed to have a limited communication range and bandwidth and the communication topology in the swarm must be implemented with a wireless mesh network. In this type of topology, each agent is a relay in the network and the data is transferred to the related destination with the help of route tables. This makes it possible to have the capability of transferring low bandwidth

data through the network with multiple hops. In this work, we have implemented this topology with a table driven routing scheme known as Bellman Ford algorithm. Bellman Ford is an algorithm that computes the shortest paths between any two node in a weighted graph [37]. It has a drawback related with the routing loop problem which occurs in an event of one or more nodes in the graph are lost during the process. The probability of vanishing vertices during the execution of the algorithm is very high since the agents in the swarm have a small range of communication and have a great possibility to lose their connection with the rest of the swarm. For this reason we have implemented an extension of the Bellman Ford algorithm known as Destination-Sequenced Distance Vector Routing Protocol(DSDV) algorithm.



	Iteration			
Node	0	1	2	3
B	0	0	0	0
A	∞	1	1	1
C	∞	1	1	1
D	∞	1	1	1
E	∞	∞	2	2
G	∞	∞	∞	3
H	∞	∞	2	2
I	∞	∞	2	2

Figure 3.5: Costs for Shortest Paths to Each Nodes from Agent 'B'

A simulation of the DSDV algorithm with a simple robot network in our problem is given in the Figure 3.5. The algorithm to calculate the shortest paths from agent 'B' to each destination on network, terminates at the end of 3 iterations. At the beginning of the process, the costs for each destinations are determined as infinitive. Then the shortest paths to the each agent in the given bidirectional graph are determined iteratively with the help of the DSDV algorithm. Since we have defined the shortest paths with the number of hops to each destination, we have used unit weight for each edge in the graph representing the cost of each hop while getting the related destination in the network is equal to 1.

We have implemented DSDV algorithm to solve the routing problem in the mesh network. Figure 3.6 shows a simulation output illustrating routing loop problem.

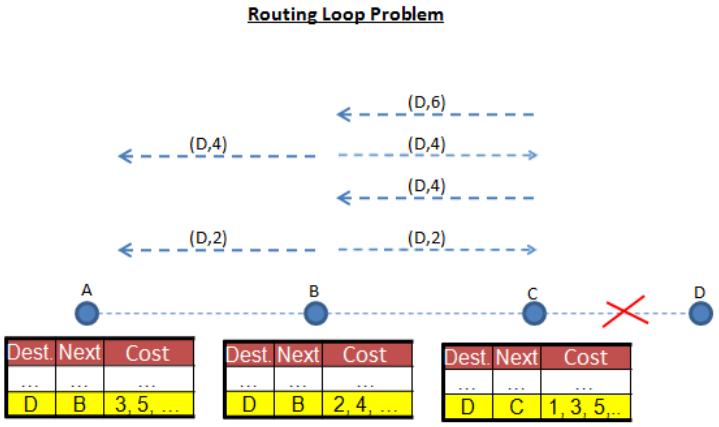


Figure 3.6: Routing Problem Engaged by a Lost of a Node in Network

Suppose that agent D has lost its contact with the network due to some malfunction or by being lost wandering outside of the communication range of its closest neighbor. Before this event, agent C had a unit distance to agent D and consequently agent B had a 2 unit distance to agent D , agent A had a 3 unit distance to agent D. In the case of a failure of agent D, on the next iteration, C updates its route table with 3 unit distance to agent D by taking as reference the agent B. Then agent B updates its route table with the shortest distance of 4 units to agent D referencing to agent C. This process diverges to infinity on the shortest paths with the increasing number of iterations. To provide a solution for this routing loop problem, DSDV algorithm implements the destination sequence numbers into the route tables of the nodes. A simulation output showing the route table for a node in a robot network is given in Figure 3.7

Route table for agent B

Destination	Next Hop	Cost	Dest. Seq. No
A	1	1	123
I	3	1	516
C	4	1	212
H	4	2	168
G	8	3	372
E	8	2	432

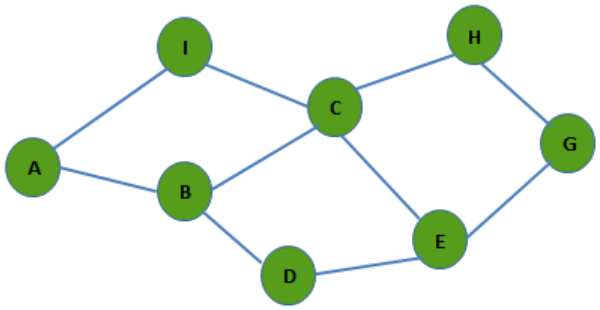


Figure 3.7: Route Table for Agent B in Robot Network

In the DSDV algorithm, each node transmits the updates in its own sequence number and routing table. In the network, when two routes to the same destination are received from two different neighbors, the algorithm observes the following rules;
- Choose the node with the larger destination sequence number
- If the sequence numbers are equal, then choose the route with minimum number of hops and update the route table.

In Figure 3.7, node B has destination sequence numbers to each destination in its route table. These numbers are used to update the network with the rules defined above, in the case of link addition and link brakes defined in following sections.

3.1.2.2 DSDV Link addition

Figure 3.8 shows a sample link addition condition to the robot network. In this example, robot A joins the network by creating a radio link with robot B. When a new node A joins the network, it transmits of its own route table including the destination to itself $< A, A, 0, 101 >$. Then the following procedure will be handled during iterations;

-Node B receives the the transmission of A and inserts a new line into its route table with $< A, A, 1, 101 >$ and propagates this new node to its neighbors

-Node C and Node D receives this transmission and inserts the new route to their route tables with $< A, B, 2, 101 >$

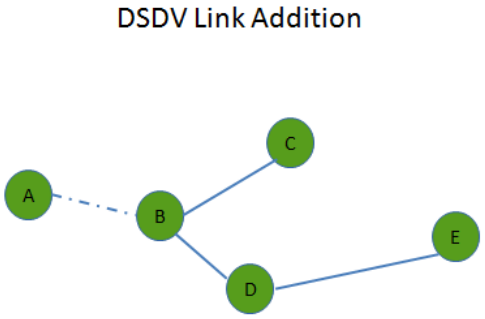


Figure 3.8: An Example for Route Table

3.1.2.3 DSDV link breaks

Figure 3.9 shows a sample link brake condition to the robot network. Robot B lost its connection with robot D. When the link between B and D breaks, node B gets no transmission from the D and notices the link breaks, then the following procedure will be handled;

- Node B update the cost for node D and E destinations to the infinity and increments the sequence numbers to these routes

- Node B propagates the updates to its neighbors and node A and node C updates the lines of the routes to the D and E, since the message from B includes higher sequence numbers for those routes.

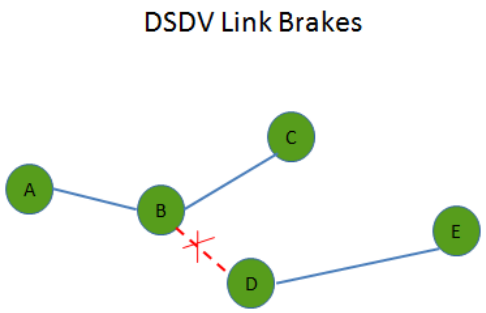


Figure 3.9: An Example for Route Table

3.1.3 Handling Lost Agents

The minimum number of position beacons required for the trilateration process is three for a two dimensional localization problem as illustrated in Section 2.3.2 . Since the agents are assumed to have a narrow communication range, it is possible to not to find three position beacons as direct neighbors for any agent at an instant time. In this case, it will not be possible to relocate these agents with trilateration and the position&velocity data will drift from the real values with the increasing time passed without trilaterions. It is important to have a solution for this problem to keep the swarm together and to increase the robustness of the system against different environmental conditions. In our algorithm, concept of 'Lost' agents and the procedures for these type of agents are described as follows:

* An agent gets into 'Lost' mode, if it doesn't find three direct neighbors of position beacons at an instant time
* If an agent is in 'Lost' mode and missed the localization process for three times, it will get into 'Return to Home' mode
* If an agent is in 'Return to Home' mode, it will directly try to reach to the center of the desired formation shape.

The idea behind the 'Return to Home' mode is basically to increase the possibility of finding position beacons which are direct neighbors of the lost agent by directing it to the center of the swarm. A simple demonstration of this procedure is illustrated in Figure 3.10

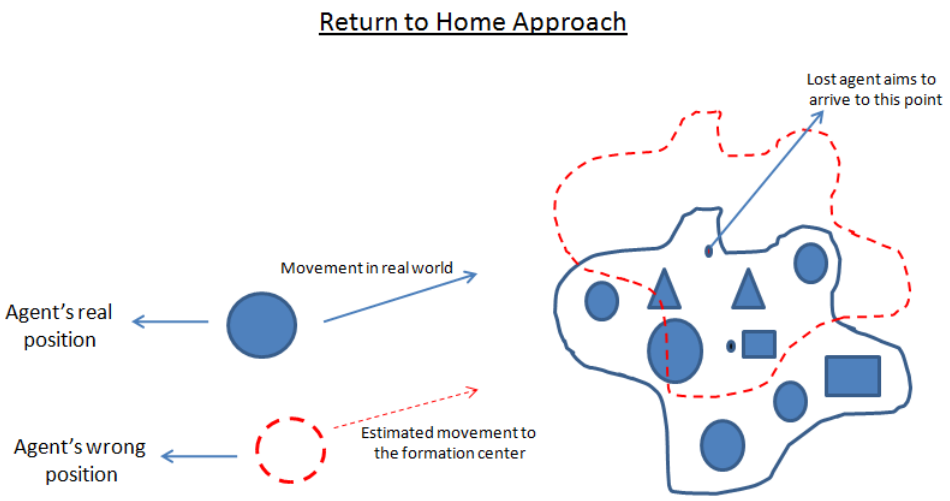


Figure 3.10: Return to Home Approach of a Lost Agent

The lost agent aims to reach to the center of the formation and due to the errors on its position&velocity data, it is expected to arrive to the red point illustrated in the figure. With this maneuver, the lost agent still have a chance to meet some position beacons in the swarm even if it directs itself to an incorrect goal state.

3.1.4 State Estimation Procedure

In local positioning subsystem, second type agents are expected to execute a state estimator algorithm in which they propagate their state vectors composed of translational position and velocities with the help of inertial measurements. As discussed at the introduction of Section 3.1 agents will update and correct their positions with the measurements provided by the trilateration process exeeceuted with the localization timer period. A Kalman estimator algorithm which uses the trilateration outputs as external measurements and the inertial measurements as inputs, is designed to fuse the sensor measurements with the trilateration calculations. The model for this observer system is defined as follows:

The state vector for each agent is defined as:

$$x_k = \begin{bmatrix} X_k \\ \dot{X}_k \end{bmatrix} \tag{3.28}$$

where X_k is the position and $\dot{X}_k$ is the velocity of the agents in x coordinates in two dimensional environment. All of the following procedures will be handled exactly the same for the state vector defined in y coordinates.

The discrete linear model is defined with the state equation of:

$$x_{k+1} = F_k x_k + B_k u_k + w_k \tag{3.29}$$

where w_k is the process noise and

$$F = \begin{bmatrix} 1 & d_t \\ 0 & 1 \end{bmatrix} \tag{3.30}$$

$$B = \begin{bmatrix} \frac{{d_t}^2}{2} \\ d_t \end{bmatrix} \tag{3.31}$$

where d_t is the propagation period and u_k is the translational acceleration measured with the help of inertial sensors. We add a random bias on the acceleration measurements on each agents acceleration input with maximum value of 0.1 mg [38]. Also, we add a white noise to the acceleration values with zero mean and 0.01 $[m/s^2]$ standart deviation [39]. The values of bias and noise terms are added to the measurements consistent with the specifications of industry grade inertial measurement units presented by the manufacturers.

The observation(external measurement) which will be calculated with the trilateration process :

$$z_k = H_k x_k + v_k \tag{3.32}$$

where v_k is the measurement noise. Since the trilateration process will provide new

position data to the agents:

$$H_k = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{3.33}$$

The noise models for the process and the measurement are modelled with:

$$w_k = \mathcal{N}(\mathbf{0}, \mathbf{Q_k}) \tag{3.34}$$

$$v_k = \mathcal{N}(\mathbf{0}, \mathbf{R_k}) \tag{3.35}$$

where w_k is the process noise with zero mean multivariate normal distribution with covariance of Q_k and v_k is the measurement noise with zero mean Gaussian distribution with a variance of R_k

The filter has two main subsections named propagate and update phases. The update phase of the filter is executed after each trilateration process. The filter equations are as follows:

Propagation phase:

$$\hat{x}_{k,k-1} = F_k \hat{x}_{k-1,k-1} + B_k u_k \tag{3.36}$$

$$P_{k,k-1} = F_k P_{k-1,k-1} F_k^T + Q_k \tag{3.37}$$

Update Phase:

$$\tilde{y}_k = z_k - H_k \hat{x}_{k,k-1} \tag{3.38}$$

$$S_k = H_k P_{k,k-1} H_k^T + R_k \tag{3.39}$$

$$K_k = P_{k,k-1} H_k^T S_k^{-1} \tag{3.40}$$

$$\hat{x}_{k,k} = \hat{x}_{k,k-1} + K_k \tilde{y}_k \tag{3.41}$$

$$P_{k,k} = (I - K_k H_k) P_{k,k-1} \tag{3.42}$$

where Q_k is the process covariance matrix and R_k is the measurement covariance chosen as

$$Q_k = \begin{bmatrix} Max.\ Acceleration\ Error * \frac{d_t^2}{2} & 0 \\ 0 & Max.\ Acceleration\ Error * d_t \end{bmatrix} \tag{3.43}$$

$$R_k = Max.\ Position\ Error\ on\ Trilateration\ Process \tag{3.44}$$

in the above equations K_k represents the Kalman gain matrix and S_k is the residual covariance of the system at time k. $\hat{x}_{k,k}$ is the posteriori state estimate updated with measurements at time k ; $\hat{x}_{k,k-1}$ is the priori estimate of the state vector predicted with inputs at time k; $P_{k,k}$ is the posteriori error covariance matrix updated with measurements at time k; $P_{k,k-1}$ is the priori estimate covariance predicted with the inputs at time k.

3.2 FORMATION CONTROL

The details of the methodology for dynamical formation control with heterogeneous mobile robots is presented in this subsection. We have implemented three different approaches as Artificial Forces method, Bubble Packing method and Randomized Fractals method. It is possible to classify these methods in two sub categories. Potential field based approaches implements Artificial Forces acting on agents to get inside and cover the desired formation shape homogeneously by avoiding collisions between the agents. The resultant positions of the agents in the formation shape is not certain, it dynamically changes with the instantaneous positions and interactions

of the agents with each other and environment. Two other methods , shape partitioning based approaches which are Bubble Packing and Randomized Fractal methods, share a common structural basis. In these approaches the complex formation shape is partitioned into goal states to cover the shape homogeneously with the mobile robots. The assignment of the agents to these goal states is handled with special algorithms to optimize the total displacements of the agents in the environment. The difference between these two methods is the partitioning approach of the formation shape. Figure 3.11 illustrates different methodologies implemented in this project.

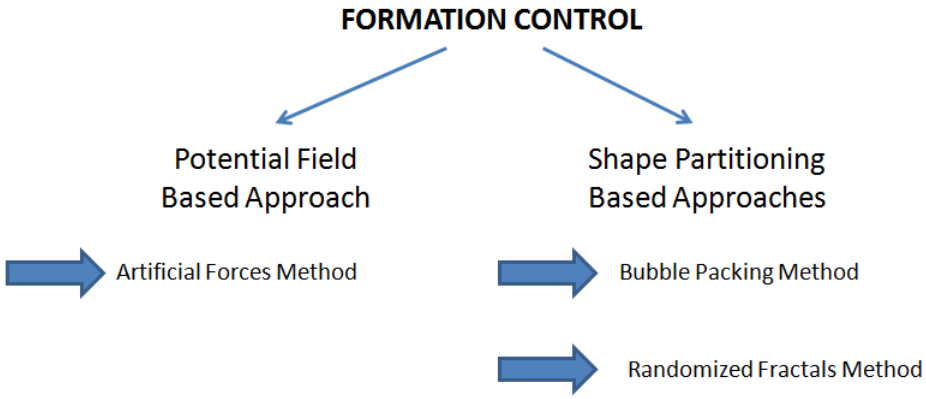


Figure 3.11: Formation Control Topologies

3.2.1 Potential Field Based Approach

3.2.1.1 Artificial Forces Method

In Artificial Forces method we have implemented potential fields over each agent arised from the interactions between agents, formation shape and environment. The positions of the agents at the formation shape are determined with local equilibrium of the swarm in which every agent is at balance under the total force acting from the environment. There are basically three different kinds of artificial forces named intermember forces representing the forces created by the other agents in the swarm to achieve collision avoidance, the attractive forces representing the forces created by the desired formation shape to attract the agent into the shape and repulsive forces created by the formation shape to keep agents inside the desired formation shape. It is

possible to augment these type of forces for specific tasks and objectives. For example, obstacle forces created by the obstacles in the environment can be implemented to achieve obstacle avoidance. Since the method to calculate the artificial forces involves contour integrals, it will be useful to give mathematical definition of contour integrals.

Consider a curve C which is a set of points $z = (x, y)$ in the complex plane defined by [40]

$$x = x(t),\ y = y(t),\ a \leq t \leq b \tag{3.45}$$

where $x(t)$ and $y(t)$ are continuous functions of the real parameter t. It is possible to write

$$z(t) = x(t) + iy(t),\ a \leq t \leq b \tag{3.46}$$

This curve is called smooth if $z(t)$ has continuous derivative of $z'(t) \neq 0$ for all points along the curve, and it is called simple if it does not cross itself as defined in Equation 3.47

$$z(t_1) \neq z(t_2)\ whenever \quad t_1 \neq t_2 \tag{3.47}$$

On the other hand if $z(a) = z(b)$ is the only intersection point, the curve is said to be simple closed curve [40]. Regarding with these given definitions, an example for a simple smooth closed curve is illustrated in Figure 3.12

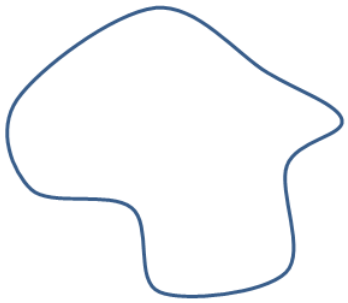


Figure 3.12: A Simple Closed Curve

Let $f(z)$ is a complex function in a domain D in the complex plane and let C be simple closed contour contained in D with initial point z_0 and terminal point z. It is possible to take the integral of $f(z)$ along the contour C [40]

$$\oint_C f(z)dz = \int_a^b f(z(t))\frac{dz(t)}{dt}dt \tag{3.48}$$

where

$$\frac{dz(t)}{dt} = \frac{dx(t)}{dt} + i\frac{dy(t)}{dt},\ a \leq t \leq b \tag{3.49}$$

To simplify this equation, one can write $f(z) = u(x,y) + iv(x,y)$ and $dz = dx + idy$ into the statements,

$$\begin{aligned}\oint_C f(z)dz &= \oint_C udx - vdy + i\oint_C udy + vdx \\ &= \int_a^b \left[u(x(t),y(t))\frac{dx(t)}{dt} - v(x(t),y(t))\frac{dy(t)}{dt}\right]dt \\ &\quad + i\int_a^b \left[u(x(t),y(t))\frac{dy(t)}{dt} + v(x(t),y(t))\frac{dx(t)}{dt}\right]dt\end{aligned} \tag{3.50}$$

We have used this simplified form of the contour integrals in utility functions described in the following section.

Utility Functions

As it is mentioned in the Section 1.3 part, the formation shapes will be simple closed contours which cannot be identified analytically. Definitions of the utility functions given in the following section are given with continuous contour integrals which requires the analytical expression of the curve on which the integral will be taken. In our thesis work, we have modified these expressions in discrete domain to provide a solution to these type of calculations without analytical expressions of the closed curves.

1- Cauchy Winding Number:

Cauchy winding number of a curve in the plane around a given point is the number of

times that curve travels counter-clockwise around the point. Suppose C is the closed curve which is a set of points $z = (x, y)$ in the complex plane and z_i is a point to check whether it is inside of the curve, then the Cauchy winding number is [22] :

$$n(C, z_i) = \frac{1}{2\pi i} \oint_C \frac{dz}{z - z_i} \tag{3.51}$$

The winding number for agent i in the swarm,

$$n(C, z_i) = \begin{cases} 1 & \text{when member i is inside C} \\ 0 & \text{when member i is outside C} \end{cases} \tag{3.52}$$

This number is used to switch on/off some of the artificial forces while the agent is inside or outside of the formation shape. We have redefined this statement in discrete domain as:

$$n(C, z_i) = \frac{1}{2\pi i} \oint_C f(z) dz \tag{3.53}$$

where

$$f(z) = \frac{1}{z - z_i} \tag{3.54}$$

Function of $f(z)$ can be partitioned into real and complex parts as:

$$u(x, y) = real(f(z)) \;\; and \;\; v(x, y) = imag(f(z)) \tag{3.55}$$

partitioning this function as mentioned in Equation 3.50

$$\oint_C f(z) dz = \oint_C u dx - v dy + i \oint_C u dy + v dx \tag{3.56}$$

then

$$n(C, z_i) = \frac{1}{2\pi i} \left[\int_a^b \left(u \frac{dx}{dt} - v \frac{dy}{dt} \right) dt + i \int_a^b \left(u \frac{dy}{dt} + v \frac{dx}{dt} \right) dt \right] \tag{3.57}$$

Discrete contour integral representation of this equation is

$$n(C, z_i) = \frac{1}{2\pi i}\left[\sum_{k=1}^{K} A_k + i\sum_{k=1}^{K} B_k\right] \tag{3.58}$$

where

$$A_k = u(x_{k+1} - x_k) - v(y_{k+1} - y_k) \tag{3.59}$$

$$B_k = u(y_{k+1} - y_k) + v(x_{k+1} - x_k) \tag{3.60}$$

$$\|z_k - z_{k-1}\| = \|z_{k+1} - z_k\|, \ \ \forall k; \ \ k = 1, 2, ..., K \ \ when \ \ K \to \infty \tag{3.61}$$

The assumption of $K \to \infty$ makes it possible to calculate the integral of Cauchy winding number with a small error with large number of K which can be achieved by partitioning the desired formation shape into small pieces with equal $p2$ norms. This approach is used to provide representations of the contour integrals in discrete domain.

Figure 3.13 shows the test results for the grid map of an environment with a formation shape, where green dots represents the inner points of formation shape and red dots represents the outer points of the formation shape where inner points in the grid map have a Cauchy winding number of 1, and the outer points have a Cauchy winding number of 0 .

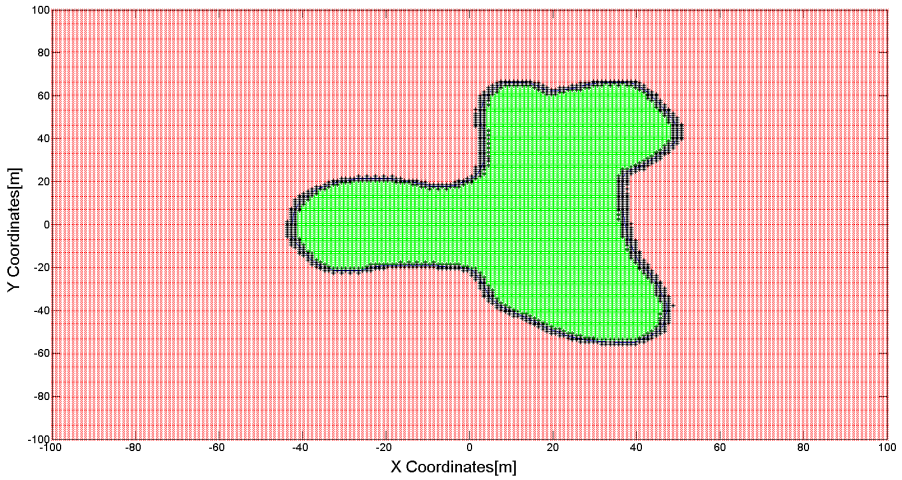


Figure 3.13: Formation Shape in an Environment

2- Length of a formation shape

The length of a formation shape can be calculated with the equation of [22];

$$l(C) = \oint_C \|dz\| \tag{3.62}$$

We have redefined this expression for this contour integral with points of $z_k = (x_k, y_k)$ in the complex plane

$$l(C) = \sum_{k=1}^{K} \sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2} \tag{3.63}$$

where

$$\|z_k - z_{k-1}\| = \|z_{k+1} - z_k\|, \ \ \forall k; \ \ k = 1, 2, ..., K \ \ when \ \ K \to \infty \tag{3.64}$$

3-Center of a Formation Shape

The center of a formation shape can be calculated with the equation of [22];

$$z_c = \frac{\oint_C z \|dz\|}{l(C)} \tag{3.65}$$

We have redefined the discrete domain expression for Equation 3.65 with points of $z_k = (x_k, y_k)$ in the complex plane

$$\begin{aligned} z_{cx} &= \frac{\sum_{k=1}^{K} x(k)}{K} \\ z_{cy} &= \frac{\sum_{k=1}^{K} y(k)}{K} \end{aligned} \tag{3.66}$$

where z_{cx} and z_{cy} are the x and y coordinates of the center of formation shape respectively and

$$\|z_k - z_{k-1}\| = \|z_{k+1} - z_k\|, \ \ \forall k; \ \ k = 1, 2, ..., K \ \ when \ \ K \to \infty \tag{3.67}$$

4- Area of a Formation Shape

Green's theorem can be used to calculate the area of a closed curve. According to this theorem, the area of D given by the double integral [41]

$$A = \int\int_D dA \tag{3.68}$$

can be calculated with the line integral of

$$A = \oint_D F ds = \frac{1}{2} \oint_D x dy - y dx \tag{3.69}$$

where

$$F(x,y) = (-y/2, x/2) \tag{3.70}$$

This contour integral can be reduced down to

$$\begin{aligned} Area &= \frac{1}{2} \oint_C x dy - \frac{1}{2} \oint_C y dx \\ &= \frac{1}{2} \int_{t=a}^{b} x(t) \frac{dy(t)}{dt} dt - \frac{1}{2} \int_{t=a}^{b} y(t) \frac{dx(t)}{dt} dt \end{aligned} \tag{3.71}$$

We have implemented the expression in Equation 3.71 in discrete domain with points of $z_k = (x_k, y_k)$ in the complex plane

$$Area = \frac{1}{2} \sum_{k=1}^{K} x_k (y_{k+1} - y_k) - \frac{1}{2} \sum_{k=1}^{K} y_k (x_{k+1} - x_k) \tag{3.72}$$

where

$$\|z_k - z_{k-1}\| = \|z_{k+1} - z_k\|, \ \ \forall k; \ \ k = 1, 2, ..., K \ \ when \ \ K \to \infty \tag{3.73}$$

Artificial Forces

Artificial forces are defined to gather the agents inside a formation shape and make

them distributed homogeneously inside the shape. It is possible to define some additional artificial forces to implement features like obstacle&collision avoidance or smooth transitions between the boundaries of the formation shape. We have implemented attractive forces, repulsive forces, intermember forces, obstacle forces and transition forces to generate individual control laws for all agents in the swarm. Summation of these different types of artificial force components define the individual control law of a single agent. Suppose the state of a member i is described by

$$X_i = \begin{bmatrix} z_i \\ \dot{z}_i \end{bmatrix} \tag{3.74}$$

where $z_i \in C$, represents the position of the i^{th} member of the swarm. The state of the whole swarm $x = \begin{bmatrix} X_1 & X_2 & ...X_n \end{bmatrix}$ is determined by the linear equations of [22]

$$\dot{x} = Ax + Bu \tag{3.75}$$

where

$$\begin{aligned} A &= diag\left(\hat{A}\right)_{nxn} \\ B &= \frac{1}{m} diag\left(\hat{B}\right)_{nxn} \end{aligned} \tag{3.76}$$

with

$$\hat{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \ \hat{B} = \begin{bmatrix} 0 & 1 \end{bmatrix} \tag{3.77}$$

The vector for individual control laws of the swarm

$$u = \begin{bmatrix} u_1 & u_2 & ... & u_n \end{bmatrix} \tag{3.78}$$

where

$$u_i = F_{i,a} + F_{i,r} + F_{i,m} + F_{i,t} \tag{3.79}$$

We have defined the concept of "coverage circle" for the agents which will be used in the artificial forces calculations. Coverage circle of agent i, C_i, is defined as the circle with minimum radius which can cover the whole agent's collision surface. The radius of this circle is given with d_c. Some of the examples of coverage circles for different types of mobile robots are illustrated in Figure 3.14 below

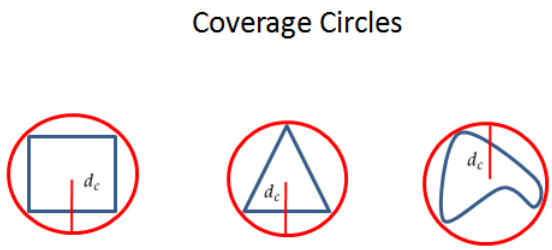


Figure 3.14: Coverage Circles of Different Types of Agents

Different artificial force components are described in details at the following section.

1- Attractive Forces

Attractive forces are the artificial force components generated by the formation shape to attract the agent towards the center of the formation .They are active when the agents are outside of the shape.

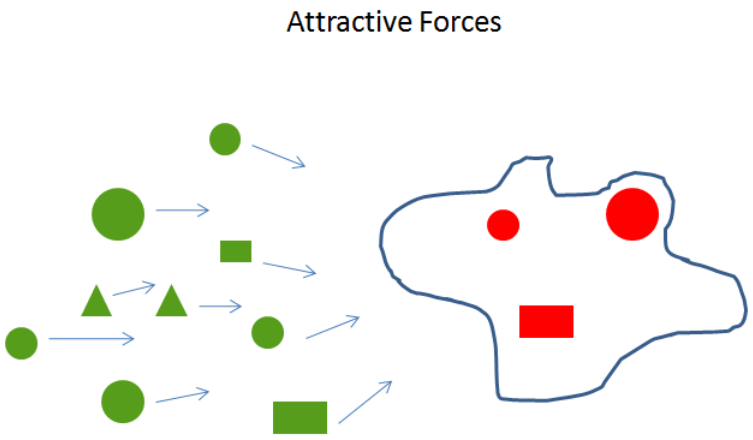


Figure 3.15: Attractive Forces Generated by the Formation Shape

The equations for the attractive forces are defined as follows in [22]:

$$F_{i,a} := \frac{k_a(1-n(C,\alpha X_i))}{l(C)} \oint_C (z-\alpha X_i)\,\|dz\| \tag{3.80}$$

where k_a is the variable gain and $\alpha = \begin{bmatrix} 1 & 0 \end{bmatrix}$. Equation 3.80 takes the contour integral of the curve defined by desired formation shape. Since we don't have the analytical expression of the desired formation shape, we have redefined this expression in discrete domain. The representation of the attractive forces on agent i on $z_i = (x_i, y_i)$ with the points of $z_k = (x_k, y_k)$ of formation shape in the complex plane [22]:

$$\begin{aligned} F_{iax} &= \frac{k_a(1-n(C,\alpha X_i))}{l(C)} \sum_{k=1}^{K} (x_k - x_i) \\ F_{iay} &= \frac{k_a(1-n(C,\alpha X_i))}{l(C)} \sum_{k=1}^{K} (y_k - y_i) \end{aligned} \tag{3.81}$$

where

$$\|z_k - z_{k-1}\| = \|z_{k+1} - z_k\|, \quad \forall k; \;\; k = 1,2,...,K \;\; when \;\; K \to \infty \tag{3.82}$$

and F_{iax}, F_{iay} are the attractive force components in x, y coordinates respectively.

2- Repulsive Forces

Repulsive forces are the artificial force components generated by the formation shape to keep the agents inside the shape. They are active when the agents are inside the shape.

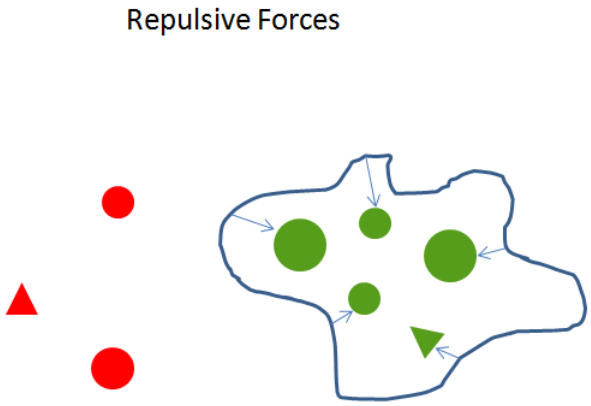


Figure 3.16: Repulsive Forces Generated by the Formation Shape

The equations for the repulsive forces are defined as follows [22]:

$$F_{i,r} := k_r n(C, \alpha X_i) \oint_C \left[\frac{\alpha X_i - z}{\|\alpha X_i - z\|^3} \right] \|dz\| \tag{3.83}$$

where k_r is the variable gain for the repulsive forces. We have redefined the Equation 3.83 in discrete domain to calculate the repulsive forces for complex shapes. The representation of the repulsive forces on agent i on $z_i = (x_i, y_i)$ with the points of $z_k = (x_k, y_k)$ of formation shape in the complex plane:

$$\begin{aligned} F_{irx} &= k_r n(C, \alpha X_i) \sum_{k=1}^{K} \frac{x_i - x_k}{\|\alpha X_i - z_k\|^3} \\ F_{iry} &= k_r n(C, \alpha X_i) \sum_{k=1}^{K} \frac{y_i - y_k}{\|\alpha X_i - z_k\|^3} \end{aligned} \tag{3.84}$$

where

$$\|z_k - z_{k-1}\| = \|z_{k+1} - z_k\|, \ \ \forall k; \ \ k = 1, 2, ..., K \ \ when \ \ K \to \infty \tag{3.85}$$

and F_{irx}, F_{iry} are the repulsive force components in x, y coordinates respectively.

3- Inter-member repulsion forces

Intermember forces are the artificial force components generated by the agents in the swarm to avoid collisions between themselves.

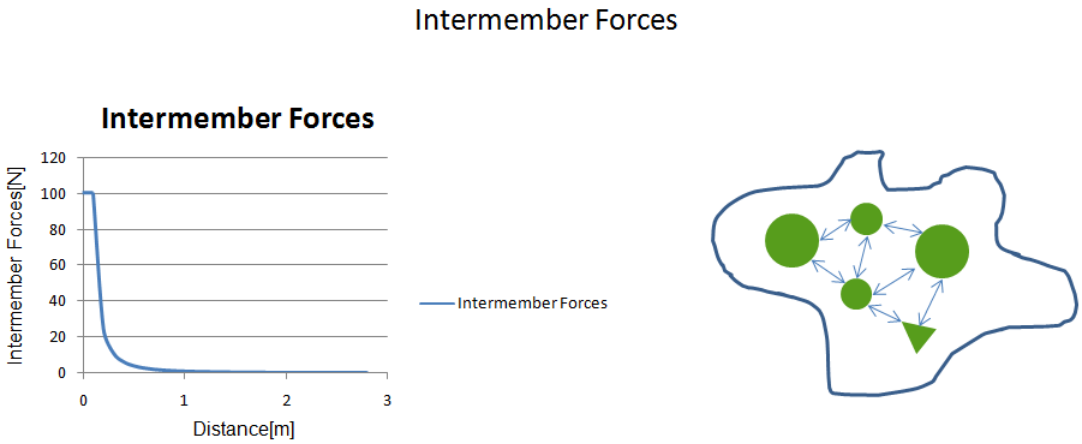


Figure 3.17: Intermember Forces Generated by Agents

The equations for the intermember forces on the agent i on $z_i = (x_i, y_i)$ in the complex plane [22],

$$F_{i,m} = k_m \sum_{j=1, j\neq i}^{N} \frac{\alpha X_i - \alpha X_j}{(\left\|\alpha X_i - \alpha X_j\right\|)} \frac{1}{(\left\|\alpha X_i - \alpha X_j\right\| - d_o)^2} \tag{3.86}$$

where d_o is the total distance between the center of coverage circles of agents which can be calculated with

$$d_o = d_{c_i} + d_{c_j} \tag{3.87}$$

The force components in x,y coordinates respectively,

$$F_{imx} = k_m \sum_{j=1, j\neq i}^{N} \frac{x_i - x_j}{\left\|\alpha X_i - \alpha X_j\right\|} \frac{1}{(\left\|\alpha X_i - \alpha X_j\right\| - d_o)^2}$$
$$F_{imy} = k_m \sum_{j=1, j\neq i}^{N} \frac{y_i - y_j}{\left\|\alpha X_i - \alpha X_j\right\|} \frac{1}{(\left\|\alpha X_i - \alpha X_j\right\| - d_o)^2} \tag{3.88}$$

where k_m is the variable gain for the intermember forces.

4- Transition Forces

Transition forces are the artificial force components to force the agent to get inside the formation shape when they are close to the boundaries. Since the attractive forces have a decreasing nature while the agent getting closer to the formation shape, we need to add this type of forcing function to ensure the agents to get inside the shape. Transition forces are active outside of the desired formation shape.

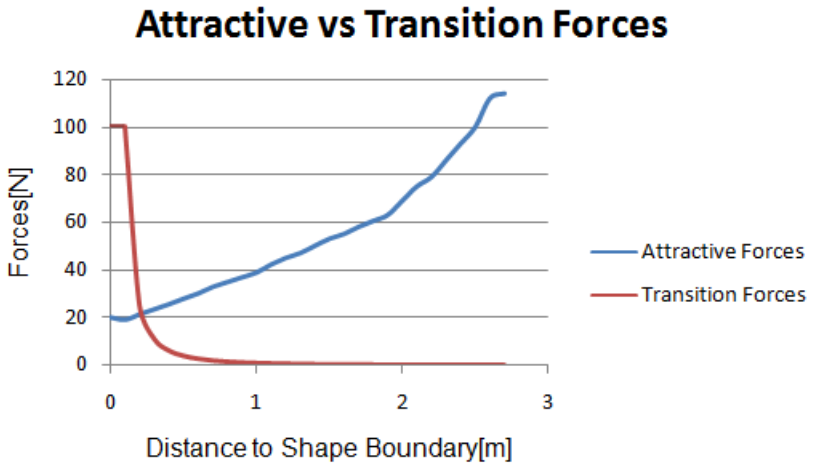


Figure 3.18: Comparison of Attractive and Transition Forces

The equation for the transition forces are defined as follows [22]:

$$F_{i,t} = k_t(1 - n(C, \alpha X_i) \oint_C \frac{z - \alpha X_i}{\|z - \alpha X_i\|} \|dz\| \tag{3.89}$$

where k_t is the variable gain for the transition forces. The representation of the transition forces on agent i on $z_i = (x_i, y_i)$ with the points of $z_k = (x_k, y_k)$ of formation

shape in the complex plane:

$$
\begin{aligned}
F_{itx} &= k_t(1-n(C,\alpha X_i)\sum_{k=1,}^{K}\frac{x_k-x_i}{\|\alpha X_i-z_k\|^3}\\
F_{ity} &= k_t(1-n(C,\alpha X_i)\sum_{k=1,}^{K}\frac{y_k-y_i}{\|\alpha X_i-z_k\|^3}
\end{aligned}
\tag{3.90}
$$

where F_{itx}, F_{ity} are the transition force components in x, y coordinates respectively and

$$
\|z_k - z_{k-1}\| = \|z_{k+1} - z_k\|, \ \ \forall k; \ \ k = 1,2,...,K \ \ when \ \ K \to \infty \tag{3.91}
$$

5- Obstacle forces

Obstacle forces are the artificial force components generated by the obstacles in the environment to achieve obstacle avoidance of the agents during formation control. The equation for the obstacle forces are defined as follows [22]:

$$
F_{i,o} := k_o \oint_O \left[\frac{\alpha X_i - z_o}{(\|\alpha X_i - z_o\| - d_{c_i})^4}\right] \|dz_o\| \tag{3.92}
$$

where k_o is the variable gain for the transition forces and d_{c_i} is the radius of coverage circle of agent i . This contour integral is taken on the curve of the obstacle with points of $z_o = (x_o, y_o)$ in the complex plane. Obstacles in the workspace are assumed to have complex shapes without mathematical definitions. We have expressed the Equation 3.92 in discrete domain. The representation of the obstacle forces on agent i on $z_i = (x_i, y_i)$ with the points of $z_{ok} = (x_{ok}, y_{ok})$ of formation shape in the complex plane:

$$
\begin{aligned}
F_{iox} &= k_o \sum_{k=1}^{K}\frac{x_i - x_{ok}}{(\|\alpha X_i - z_{ok}\| - d_{c_i})^4}\\
F_{ioy} &= k_o \sum_{k=1}^{K}\frac{y_i - y_{ok}}{(\|\alpha X_i - z_{ok}\| - d_{c_i})^4}
\end{aligned}
\tag{3.93}
$$

where

$$\|z_k - z_{k-1}\| = \|z_{k+1} - z_k\|, \ \ \forall k; \ \ k = 1,2,...,K \ \ when \ \ K \to \infty \tag{3.94}$$

Buffer Zone Implementation

The attractive and transition forces are defined to be active when the agents are outside of the shape. On the other hand, the repulsive forces are active when agents are inside the shape. Because of the sharp transitions on the total artificial force acting on an agent which is crossing the boundary of the formation shape, it is possible to have a chattering effect. To avoid this chattering effect and provide a smooth transition of the agent to pass the boundary of formation shape, we have implemented a buffer zone defined around the boundaries of the formation shape. The main approach during the transition on the boundaries is to dynamically change the variable gains of these artificial forces linearly between zero and nominal values at the boundary conditions. The variable gains of the attractive and transition force components have their nominal values at the outer boundary of shape and these gains are decreased down to zero linearly while travelling towards the inner boundary of the shape. On the other hand, the variable gain for the repulsive forces has its nominal value at the inner boundary of the shape and this gain is decreased down to zero linearly while travelling towards to outer boundary of the shape.

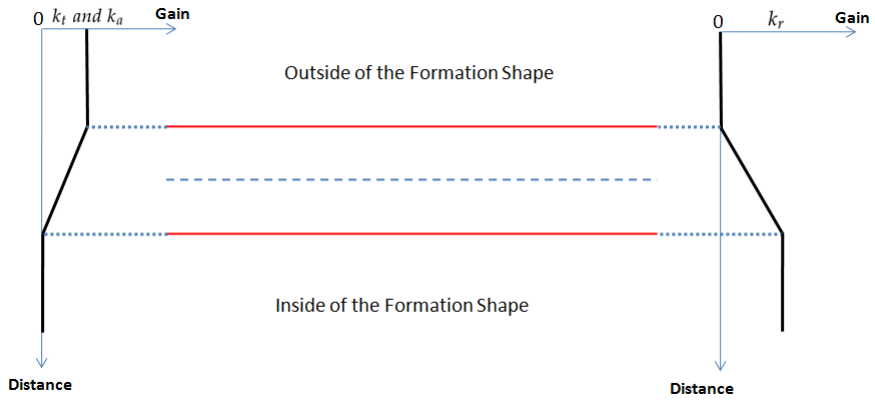


Figure 3.19: Transition of the Artificial Forces on Buffer Zone

3.2.2 Shape Partitioning Methods

In shape partitioning methods we have reduced down the formation control problem into two subproblems. The first part of the solution is to partition the desired formation shape into potential goal states according to the agent types to cover the desired formation shape homogeneously. We have proposed two different solutions to the shape partitioning problem in this thesis work, Bubble Packing method and Randomized Fractals method. The second part of the solution is the decision process to assign the agents to these goal states continuously to minimize the total displacement of swarm while achieving the desired formation shape. During this decision process, the cost of reaching different goal states is defined with the displacement on the shortest path while reaching that goal state. Our algorithm tries to reduce down these cost valus to minimize the total displacement of the swarm.

Shape partitioning methods have an approach to direct the agents to the goal states in which they have assigned dynamically. In this approach it is still needed to have collision avoidance algorithms to prevent the agents to collide with each other and workspace obstacles while travelling towards these goal states. For this purpose we add obstacle and inter-member forces to the individual control laws defined in shape partitioning approaches.

3.2.2.1 Determining the Potential Goal States

Bubble Packing Method

Bubble Packing method is widely used in mesh generation problems. It basically depends on covering a curve, surface or a volume with a proper number of bubbles by packing them tightly which mimic a Voronoi diagram, from which a set of well-shaped Delaunay triangles can be created by connecting the centers of the bubbles [28]. The algorithm places the bubbles with their initial conditions in the environment and apply them interbubble forces which imitates the Van der Waals forces between the molecular bonds to distribute the bubbles homogeneously. Here, the main idea is to generate a mesh for a surface with identical bubbles to mimic a regular Voronoi diagram with the vertices represented by the centers of these bubbles. On the other

hand, adaptive population control methods are developed to increase the number of bubbles to fill the gaps in the shape and to remove the excess bubbles which are overlapping with each other and the shape boundaries [28].

Since the numbers and the the radius of coverage circles defined at Section 3.2.1.1 are predetermined in our formation control problem, the general bubble meshing algorithm have to be adapted to meet the requirements for shape partitioning in formation control. In our project we have adapted this algorithm to represent the agents in the swarm as bubbles with the radius of their coverage circles and create a mesh by using these bubbles. The details of the implementation are given in following sections.

I - Initial Placements of the Bubbles

The initial bubble placements are important because it will greatly reduce the convergence time of the partitioning process. In this work, we have initiated the bubbles close to the center of the formation shape randomly. The implemented algorithm for initial bubble placements is provided as follows:

Data: Set of Bubbles, Desired Formation Shape
Result: Initial Placements of the Bubbles
Initialize free configuration space C_{free} as the desired formation shape
for <*Each Bubble i*> **do**
    *Calculate the free configuration space,C_{free}, for bubble *i*;
    *Put the Bubble *i* to the closest point to formation center z_c in the free configuration space;
    *Add the agent *i* into the configuration space as an obstacle ;
end

Algorithm 1: INITIALIZE_BUBBLE_POSITIONS

The term free configuration space C_{free} will be analyzed in detail in Section 3.2.2.2. A simulation output, with a sample formation shape and 11 coverage circles which are representing a heterogeneous swarm composed of agents from 3 different size is illustrated in Figure 3.20 below.

Figure 3.20: Initialization of the Bubble Packing Algorithm

II- Bubble Meshing Process

The bubbles are distributed homogeneously with this process under two kinds of forces, interbubble forces and shape repulsive forces. The interbubble forces are proximity-based forces so that a system of bubbles is in equilibrium when bubbles are distributed over the whole formation shape. The implemented force equation is given

$$f_i(l) = \begin{cases} al^3 + bl^2 + cl + d & \text{when } 0 \leq l \leq l_0 \\ 0 & l > l_0 \end{cases} \tag{3.95}$$

where l is the distance between the centers of the related bubbles and a, b, c, d and l_0 are the variables to tune the force acting on the bubbles.

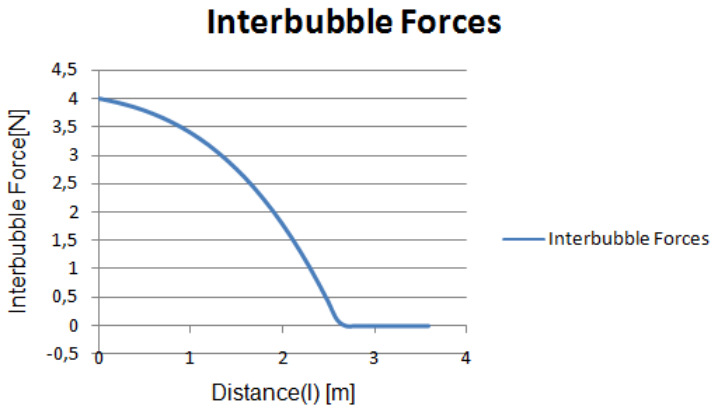


Figure 3.21: Interbubble Forces

The shape repulsive forces have the same characteristics with the repulsive artificial forces discussed in Section 3.2.1.1. The representation of shape repulsive forces for the desired formation shape C with the points of $z_k = (x_k, y_k)$ the complex plane:

$$f_r(X_i) := \oint_C \left[\frac{\alpha X_i - z}{\|\alpha X_i - z\|^3} \right] \|dz\| \tag{3.96}$$

where k_r is the variable gain for the shape repulsive forces. We have defined the shape repulsive forces on agent i in discrete domain with the points of $z_k = (x_k, y_k)$ on formation shape in the complex plane:

$$f_r(X_i) = \sum_{k=1}^{K} \frac{\alpha X_i - Z_k}{\|\alpha X_i - z_k\|^3} \tag{3.97}$$

where

$$\|z_k - z_{k-1}\| = \|z_{k+1} - z_k\|, \ \ \forall k; \ \ k = 1, 2, ..., K \ \ when \ \ K \to \infty \tag{3.98}$$

The bubbles are distributed homogeneously under the influence of these two forces and they get an equilibrium state in which the total net forces acting on individual bubbles reaches zero. Figure 3.22 shows a simulation output of Bubble Packing algoithm which is initialized with coverage circles derived from the output of the INITIALIZE_BUBBLE_POSITIONS algorithm. The coverage circles are homogeneously distributed in the formation shape with the help of inter-bubble and shape repulsive forces. The final equilibrium states of the bubbles determine the potential goal states $g_i \in G$ of the agents in the swarm to cover the formation shape.

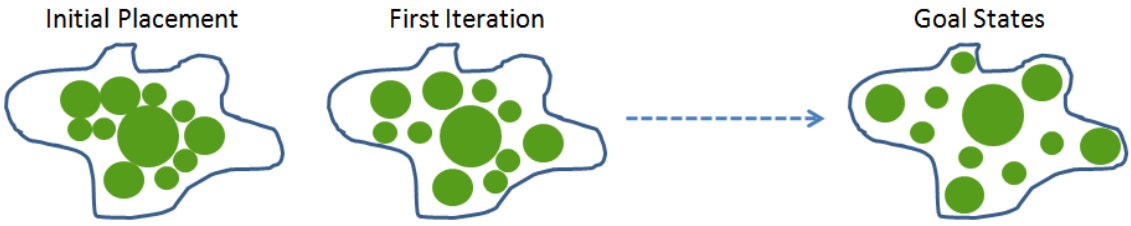


Figure 3.22: Bubble Packing Algorithm

Randomized Fractals Method

Randomized fractals methods are used to cover surfaces or volumes randomly with

fractals. The main idea is to fill the shapes iteratively with fractals which has the areas determined by the rule of [27] :

$$A_i = \frac{A}{\zeta(c,N)(i+N)^c} \tag{3.99}$$

where A is the total area to cover and A_i is the area of the i^{th} fractal. The parameters of c and N can be chosen to implement different changes on the fractals' areas with the increasing number of iterations with $c > 1$ and $N > 0$. Here ζ is the Hurwitz function defined by

$$\zeta(c,N) = \sum_{i=0}^{\infty} \frac{1}{(i+N)^c} \tag{3.100}$$

It is possible to write,

$$\sum_{i=0}^{\infty} A_i = \sum_{i=0}^{\infty} \left(\frac{A}{\zeta(c,N)(i+N)^c} \right) \tag{3.101}$$

which tells us the sum of the all areas A_i is the total area of A and the algorithm is space filling. This approach implements the fractals infinitely by reducing the areas in accordance with the Equation 3.99, to cover the desired formation shape randomly.

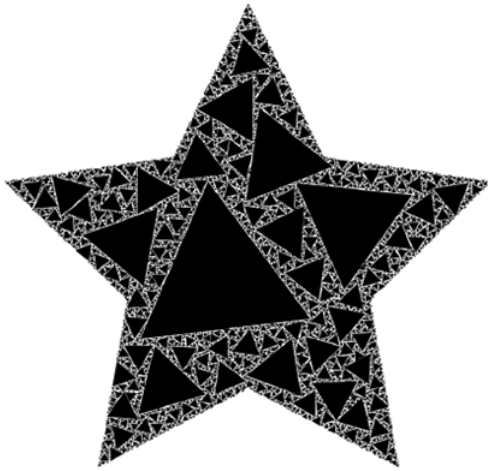


Figure 3.23: Randomized Fractals [27]

Types and the number of the agents, which will be represented with their coverage circles, are predetermined in our work. By using this constraint, we have adapted the randomized fractals algorithm as follows:

Data: Set of Coverage Circles, Desired Formation Shape
Result: Potential Goal States
Initialize free configuration space C_{free} as the desired formation shape
for <*Each Coverage Circle i*> **do**
    *Calculate the free configuration space C_{free} for circle i;
    if $C_{free} \neq Full$ **then**
        *Put the circle i randomly in C_{free} ;
        *Add the agent i into configuration space as an obstacle ;
    else
        *Break and warn the operator to increase the size of formation shape
    end
end

Algorithm 2: RANDOMIZED_FRACTALS_ALGORITHM

The algorithm iteratively checks the free configuration space that if there is enough space. If so, it inserts a coverage circle which is representing the current agent in the swarm, randomly in the desired formation shape. At the end of each insertion, it updates the free configuration space and jumps to next iteration. When algorithm terminates, the positions of the coverage circles which are randomly placed in desired formation shape gives us the goal states $g_i \in G$ of the agents.

Figure 3.24: Simulation Output of the Algorithm

Figure 3.24 shows a simulation output of the Randomized Fractals algorithm. In this figure, coverage circles are representing the members of a swarm which consist of 11 heterogeneous agents from 3 different sizes. These coverage circles are randomly

distributed in the desired formation shape with the help of Randomized Fractals algorithm. Positions of the coverage circles define the goal states $g_i \in G$ of the agents in the formation shape.

3.2.2.2 Decision Process on Goal States

In Section 3.2.2.1, we have reduced down the formation control problem in which every agent is expected to decide where to position in a given set of possible goal states $g_i \in G$. During this decision process, the cost of reaching different goal states will be the main criteria for each agent. Given goal states and cost values to these goal states, each agent must decide where to position in the formation. This process must be held to optimize the utility of every agent with a collaboration. It is obvious that some of the agents may want to choose the same goal point to reach, so the swarm must reach a global consensus on target points and cases including conflictions must be handled. Our main approach to provide a solution for this problem is to make each agent to calculate the costs of its own to reach the goal states $g_i \in G$ and then reach a global consensus with the other agents to minimize the overall displacements of the swarm.

The algorithm which handles the assignment process of the agents to the goal states, is implemented in four stages. At the first stage, each agent calculates its own free configuration space. Secondly, agents calculate their visibility graphs by using their free configuration space. At the third stage, agents calculate and report their costs to reach each of the goal states $g_i \in G$ with the help of Dijkstra's algorithm. These cost values are defined as the displacement to reach a goal state on shortest path. Finally, agents are assigned to the goal states with the help of Hungarian algorithm which uses the cost values reported by the agents. Hungarian algorithm handles the assignment process to minimize the overall cost(i.e. total displacement) of the swarm.

Free Configuration Space

We have defined the shortest paths to the goal states of $g_i \in G$ in the free configuration space to avoid collisions with the workspace obstacles [42]. In our implementation,

each agent calculates its free configuration space by extracting the forbidden space from the configuration space itself. Forbidden space is calculated by augmenting the workspace obstacles with the help of Minkowski sums and we have implemented this process as follows.

Assume an environment with set of obstacles $S = \left\{P_1, P_2, ..P_t\right\}$. Configuration for agent i can be described with the position of the center of its coverage circle with $R = \left\{x_i, y_i\right\}$. Configuration space of i^{th} agent is the workspace itself and represented by $C(R_i)$. This configuration space is composed of two subspaces; free configuration space and forbidden configuration space of agent i [42].

$$C(R_i) = C_{free}(R_i, S) + C_{forb}(R_i, S) \tag{3.102}$$

In our implementation each agent calculates its forbidden space by augmenting the workspace obstacles with the help of Minkowski sum method. Minkowski sum method is implemented as follows:

Let a single obstacle is described with a point set of S_1 and the agent is described with a point set of S_2. The Minkowski sum of these two sets $S_1 \subset R^2$ and $S_2 \subset R^2$ can be calculated with the following [42],

$$S_1 \oplus S_2 := \left\{p + q : p \subset S_1, q \subset S_2\right\} \tag{3.103}$$

where $p + q$ denotes the vector sum of the vectors p and q.

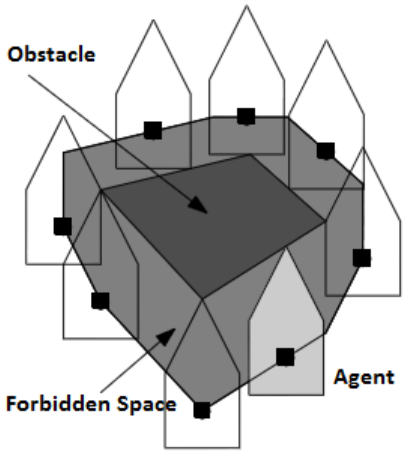


Figure 3.25: Forbidden Space Caused by an Obstacle [42]

Figure 3.25 shows a forbidden space related with an obstacle. Forbidden space for agent i, $C_{forb}(R_i, S)$, is the sum of the forbidden spaces calculated for each obstacle in the environment [42]. Agents extract their forbidden space $C_{forb}(R_i, S)$ from the configuration space itself to calculate their free configuration spaces.

Visibility Graphs and Dijkstra′s Algorithm

We have provided collision avoidance while travelling towards to the goal states $g_i \in G$ by defining the shortest paths in the free configuration spaces of the agents. In [42], an additional constraint for the shortest path is defined as follows :

> The shortest path between p_{start} and p_{goal} among a set S of augmented polygonal obstacles consists of the arcs of the visibility graph $\gamma_{vis}(S^*)$ where $S^* := S \cup \{p_{start}, p_{goal}\}$

A visibility graph, $\gamma_{vis}(S^*)$, is a graph which is set of interior nodes representing the vertices of the set of obstacles, S, in the environment and edges which represents visible (which are not crossing and interior region of an obstacle) connections between these nodes [42].

In this project we have updated this constraint in our algorithm. We have inserted all of the goal states $g_i \in G$ in the visibility graphs of each agent to calculate the shortest paths to all of these goal states rather than a specific p_{start} and p_{goal} points given in

the definition. In the implementation phase, we have used the obstacles which are augmented with the Minkowski sums to calculate the free configuration space. Let these set of augmented polygonal obstacles represented with $S_i \subset S$.

Algorithm to calculate the visibility graph of $S^* := S \cup \left\{ g_i \in G \right\}$

Data: Set of Vertices Included in S^*

Result: Visibility Graph of S^*

Initialize a graph $\gamma = (V, E)$ where V is the set of al vertices of the polygons in S^* and $E = \oslash$

for <*all vertices* $v \subset V$> **do**

    W = VISIBLE_VERTICES(v,S);

    Add edges W to list E;

end

Algorithm 3: VISIBILITY_GRAPH

where $VISIBLE\_VERTICES(v, S)$ algorithm checks whether line segments drawn from v to all vertices in S is intersecting an interior area of any obstacle in the environment and returns the non-intersecting edges. With the help of this $VISIBILITY\_GRAPH$ algorithm $SHORTEST\_PATH$ algorithm can be defined as follows.

Data: A set S of disjoint polygonal obstacles, position of agent, p_{start}, and all goal states $g_i \in G$

Result: The shortest collision-free paths from p_{start} to all goal states $g_i \in G$

* Assign γ = VISIBILITY_GRAPH(S^*)

* Assign each arc (v, w) in γ_{vis} a weight, which is the Euclidian length of segment vw

for <*Each goal state* $g_i \in G$> **do**

    * Use Dijkstra's algorithm to compute a shortest path between p_{start} and $g_i \in G$ in γ_{vis}

end

Algorithm 4: SHORTEST_PATH

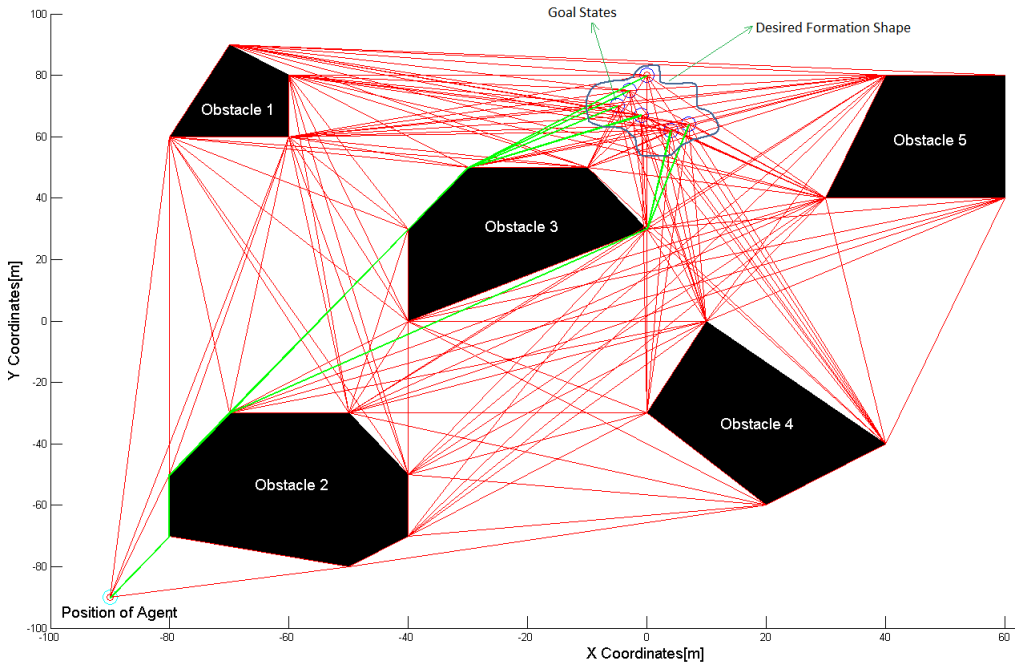


Figure 3.26: Shorthest Paths to Goal States $g_i \in G$ in a Visibility Graph

Figure 3.26 shows a simulation output executed with 5 workspace obstacles and 6 goal states in desired formation shape. In this algorithm, agent first calculates its own visibility graph by adding goal states $g_i \in G$ in graph as nodes. The visibility graph is illustrated with red edges in the figure. Then it calculates the shorthest paths to the goal states, which are given with green edges in the figure. We have used Dijkstra algorithm to compute the shortest path between two nodes in graph with multiple edges, each having a non-negative weight. In our work, the weights of the edges in γ_{vis} , are calculated with the Euler distance between nodes in the workspace. Dijkstra algorithm is a tree search algorithm and time complexity of the original algorithm is $O(n^2)$ where n is the number of the nodes in the graph. With the usage self balancing binary search tree, the algorithm requires $O(k+nlogn)$ time in the worst case where k is the number of edges in the graph. The algorithm for the Dijkstra's is implemented as follows [42]:

Data: γ_{vis} , Position of the agent "*source_node*"

Result: Shortest Distance to goal states $g_i \in G$ from the *source_node* in γ_{vis}

for <*Each Vertex* $v \subset \gamma_{vis}$> **do**

    Distance[v] := ∞ ;

    Previous[v] := *undefined* ;

end

Distance[*source_node*] :=0 ;

Q:= The set of all vertices $g_i \subset \gamma_{vis}$;

while <$Q \neq null$> **do**

    u:= Node in Q with smallest distance to *source_node*;

    remove u from Q;

    for <*Each Neighbor v of u*> **do**

        alt:= Distance[u] + Cost Between u and v nodes;

        if *alt*<*Distance[v]* **then**

            Distance[v] :=alt;

            Previous[v] :=u;

        end

    end

end

return Previous[v];

Algorithm 5: DIJKSTRA_ALGORITHM

In algorithm above $Distance[x]$ function call, calculates the total cost from the *source_node* to the x vertex, and $Previous[x]$ function call, returns the previous node in optimal path from *source_node* . This algorithm calculates the shortest paths from the position of an agent to all available goal states. These cost values are used to assign the agents to the goal states by minimizing the total displacement.

Collaborative Decision Process of Final Goal States

We have provided an algorithm to calculate the costs to the goal states $g_i \in G$ with the help of Visibility Graphs and Dijkstra's algorithm in previous sections. These costs are defined as displacements on the shortest paths to the goal states in the visibility graphs for each agent. In this project, our aim is to minimize the total displacement

of the individuals while achieving the desired formation shape. For this purpose we have implemented an algorithm to minimize the overall displacement of whole swarm while achieving a formation shape. The problem related with this process can be defined as follows:

We have n number of agents in our swarm and n number of goal states $g_i \in G$ placed in desired formation shape. Each agent has reported their costs (i.e. minimum displacements) to reach each of these goal states and we have to implement an algorithm to assign the agents to these goal states while minimizing the total displacement of the swarm. This is a generalized assignment problem and we have used Hungarian algorithm which is a combinational optimization algorithm that solves this assignment problem. To implement this algorithm, a complete bipartite graph $G = (S,T,E)$ with $n \in S$ agents and $g_i \in T$ goal states is constructed.In this graph, each agent has a cost which is defined by the shortest path to the destination in the workspace for different goal points.

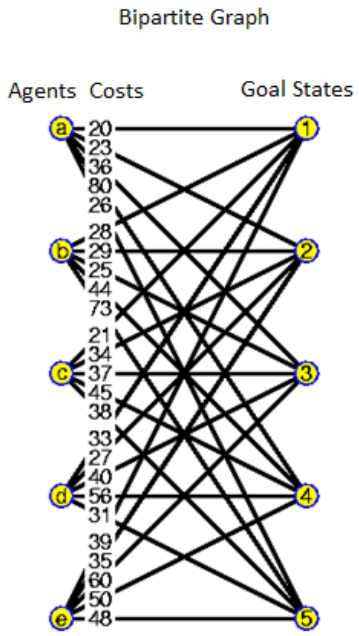


Figure 3.27: Sample Bipartite Graph Used in Assignment Problem [43]

We have defiend a cost matrix C to implement the Hungarian algorithm. The dimensions of the cost matrix is nxm in which each element represents the cost of assigning

the goal state *m* to the agent *n*. Since we have equal number of agents and goal states, the cost matrix will be a square *nxn* matrix. The algorithm for the assignment process as follows:

Data: Cost Matrix , *C*

Result: Assignment Array of Agents to Goal States

Label1 ;

for *Each Row, R, in C* **do**
| Find the smallest element and subtract it from every element

end

Label 2 ;

if *A column,K, contains more than one zero* **then**
| Repeat Label1 for each column, *K*

end

Label 3 ;

Select element in columns for which a distinct minimum weight has been determined and add to solution

Label 4 ;

If it is not possible to reach the full solution, flag rows without solutions. Flag all columns in flagged rows that contain a zero. Flag all rows with a previously determined solution in previously flagged columns.

Label 5 ;

From elements remaining in flagged rows and unflagged rows, determine the element which has smallest value and assign this value to γ. Subtract γ from every unflagged element and add γ to every element that has been flagged twice.

Label6 ;

Goto Label3 until full solution has been achieved.

Algorithm 6: HUNGARIAN_ALGORITHM

Hungarian algorithm returns a vector of *nx*1 and each row in this vector represents the goal state that the related agent is assigned. With this assignment, the total cost (i.e. total displacement) of the swarm is minimized.

3.2.3 Mesh Quality Measurement

We have provided two different solutions to the shape partitioning problem. Bubble Packing method partitions the desired formation shape into goal states by distributing the bubbles which are representing our agents in the swarm. On the other hand, Randomized Fractals method randomly places the fractals in the desired shape and these fractals are representing our agents in the swarm. In this project one of our aims is to distribute the agents in the formation shape homogeneously. So it is useful to define a metric to compare the performances of these algorithms, i.e. a metric which represents how these methods homogeneously distribute the agents in the formation shape. One of the criterias we have used is topological mesh irregularity [28] which is commonly used by mesh generation problems. To measure the topological mesh irregularity in our swarm, first we have constructed a Voronoi diagram with the nodes which are the center positions of the agents' coverage circles. Figure 3.28 shows a simulation output about the final states of the agents in the desired formation shape. Centers of these coverage circles are used as positions of nodes while constructing the Voronoi diagram as illustrated in the right hand side of the figure.

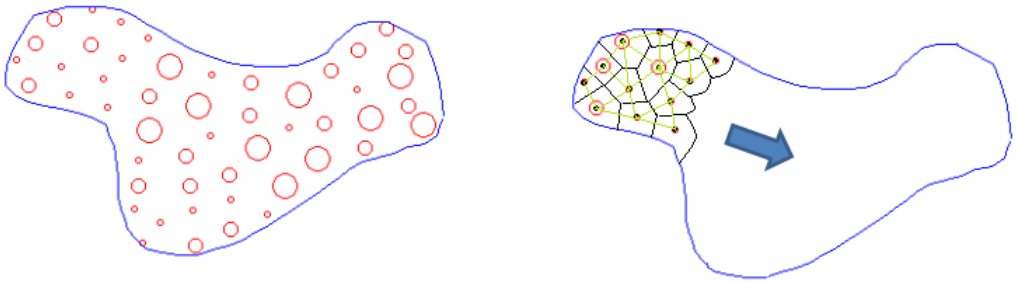


Figure 3.28: Voronoi Diagram Constructed with Final Positions of Agents

With the help of this Voronoi diagram, the topological mesh irregularity is defined by:

$$\varepsilon_t = \frac{1}{n}\sum_{i=0}^{n} |\gamma_i - D| \tag{3.104}$$

where

$$D = \begin{cases} 6 & \text{for triangles in Voronoi Diagram} \\ 12 & \text{for tetrahedras in Voronoi Diagram} \end{cases} \tag{3.105}$$

and γ_i represents the degree, or the number of neighboring nodes connected to the i^{th} interior node, and n represents the total number of interior nodes, i.e. the number of bubbles or fractals. In this implementation, topological mesh irregularity vanishes when all nodes have D neighbors, but this ideal case is almost not possible in practical applications. Figure 3.29 shows a node with 6 neighbors in an example Voronoi diagram.

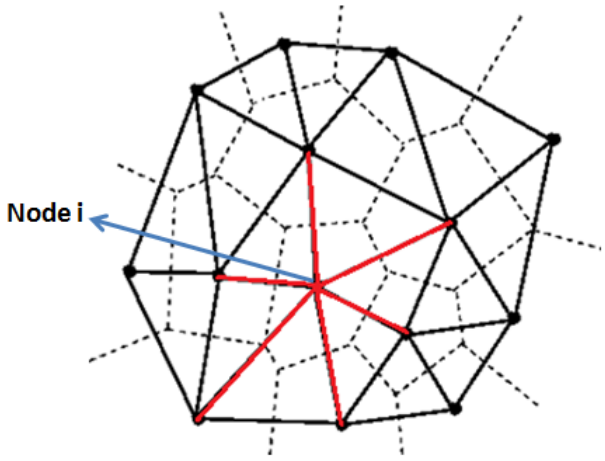


Figure 3.29: A Node with 6 Neighbors

Another metric we have used is geometrical mesh irregularity [28] With the help of the Voronoi diagram generated for the final state of the swarm, this irregularity is defined as [28]:

$$\varepsilon_g = \frac{1}{m}\sum_{i=0}^{m} (A - \frac{r_i}{R_i}) \tag{3.106}$$

where

$$A = \begin{cases} 0.5 & \text{for triangles in Voronoi Diagram} \\ \sqrt{2/11} & \text{for tetrahedras in Voronoi Diagram} \end{cases} \tag{3.107}$$

and m represents the number of nodes, r_i is the radii of inscribed circle of Voronoi cell belonging to node i and R_i is the radii of the circumscribing circle of Voronoi cell belonging to node i. Figure 3.30 shows inscribing and circumscribing circles of a Voronoi cell.

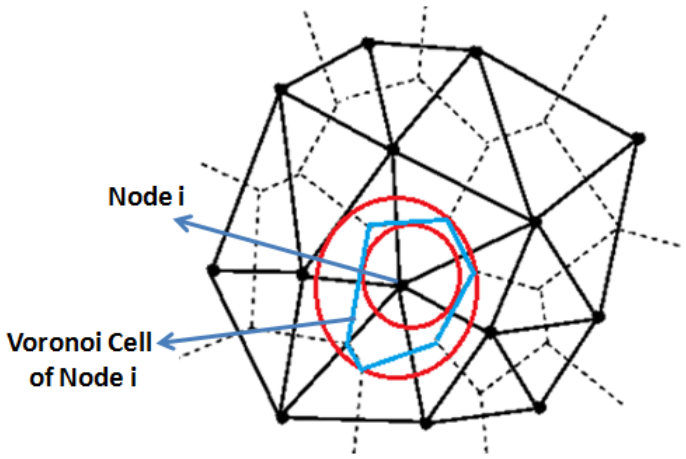


Figure 3.30: Inscribing and Circumscribing Circle of a Voronoi Cell of Node i

These irregularities show that how homogeneously the agents cover the desired formation shape in their final states. We expect to have lower values of irregularities for more homogeneous distributions.

3.2.4 Control System Design for Individual Agents

In shape partitioning methods we have implemented algorithms to calculate the potential goal states in desired formation shape. Then we have defined the assignment procedure of the agents to these goal states to minimize the total displacement of the agents in previous chapters. At this point, it is needed to implement navigation control laws for individual agents to make them reached to the goal states they have assigned.

Since the environment is dynamically changing with lots of mobile agents, it is very probable to have different assignments to these goal states at each execution step of formation control algorithm. Control system must react to these dynamically changing setpoints. For these purposes, we have designed a velocity controller with a large bandwidth as the inner loop of the control system and the outer loop is composed with a velocity setpoint generator. The block diagram for the proposed controller structure is presented in Figure 3.31.

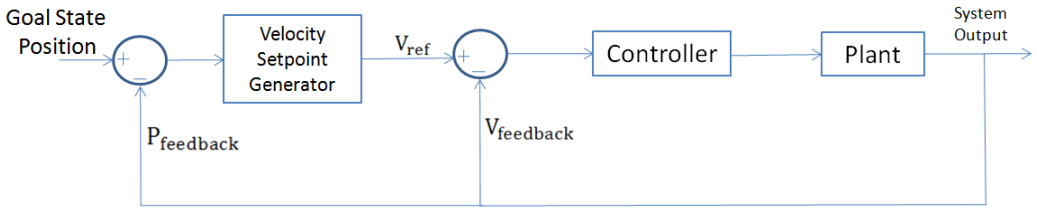


Figure 3.31: General Scheme of the Control System

In our implementation, velocity setpoint generator provides instant setpoints for the inner loop based on the current position of the agent and the desired goal state position. This loop calculates the amplitude of the velocity setpoint proportional with the euclidian distance of agent to the desired goal state. The direction of this velocity setpoint vector has a bearing angle of the line segment drawn from the agent to the goal state. The additional terms discussed in Section 3.2.2 related with collision avoidance is added to this velocity setpoint. This generator saturates the amplitude of the setpoint vector with 0.5 [m/sec] to prevent the agents' travelling in the environment so fast. Figure 3.32 shows the amplitude and the bearing angle to the destination goal state used by the velocity setpoint generator.

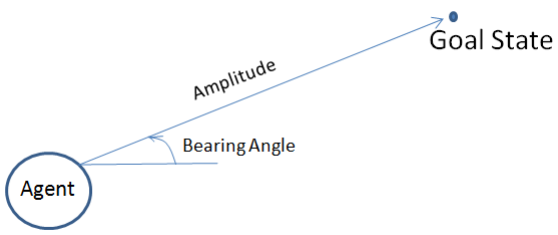


Figure 3.32: Velocity Setpoint Generation

The inner loop is designed to track the velocity setpoint provided by the generator and it has a state feedback structure. The gains for the feedback states are calculated with LQR methodology. A simple mass, damper type second order linear system is used to provide a linear model for the controller design. The translational friction force is assumed to be linear with the velocity of each agent. Different mass and friction coefficients for heterogeneous mobile robots are used in control system design. To track the desired velocity setpoint, the model of the system is augmented with an artificial error state e,

$$\begin{bmatrix} \dot{v} \\ \dot{e} \end{bmatrix} = \begin{bmatrix} -b/m & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v \\ e \end{bmatrix} + \begin{bmatrix} 1/m \\ 0 \end{bmatrix} F_{net} \quad and \quad y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} v \\ e \end{bmatrix} \tag{3.108}$$

where b is the linear friction force coefficient and m is the mass, v is the linear velocity of the agent and e is the augmented error state which is the integral of the velocity state. Here the v state is the stabilizing part of the controller and the e error state is the tracker part of the controller. The state feedback gain, K, which minimizes the quadratic cost function of

$$J = \int_{t_o}^{t_1} (x^T Q x + u^T R u) dt \tag{3.109}$$

is calculated with help of $'lqr'$ function of MATLAB for the given system in Equation 3.108 with the gain matrices of

$$Q = \begin{bmatrix} q_1 & 0 \\ 0 & q_2 \end{bmatrix} \quad and \quad R = r_1 \tag{3.110}$$

The parameters for the controller design are determined with the approach presented below,

$$q_1 = \frac{1}{t_1 (x_{1max})^2}; \; q_2 = \frac{1}{t_2 (x_{2max})^2}; \; and \; r_1 = \frac{1}{(u_{1max})^2}; \tag{3.111}$$

Here t_i is the desired settling time for x_i which is determined as 1.5 seconds for the velocity and 0.01 seconds for the integral state. The statement of x_{imax} represents the

expected maximum value of the state i , which is defined 1 [m/sec] for the velocity state and 4 [m] for the error state. u_{1max} represent the maximum allowable input signal which is defined as 3 [N]. The structure of the inner loop is illustrated in Figure 3.33

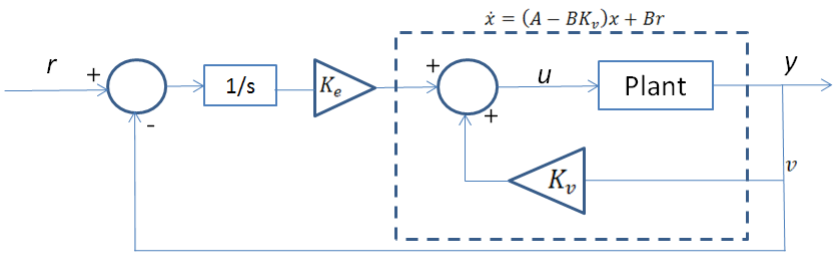


Figure 3.33: Inner Loop Structure

The error state ([m]) between the reference and the velocity feedback is integrated and multiplied with the error gain, and the velocity state is multiplied with the velocity gain. These two components generate the total control input of the system.

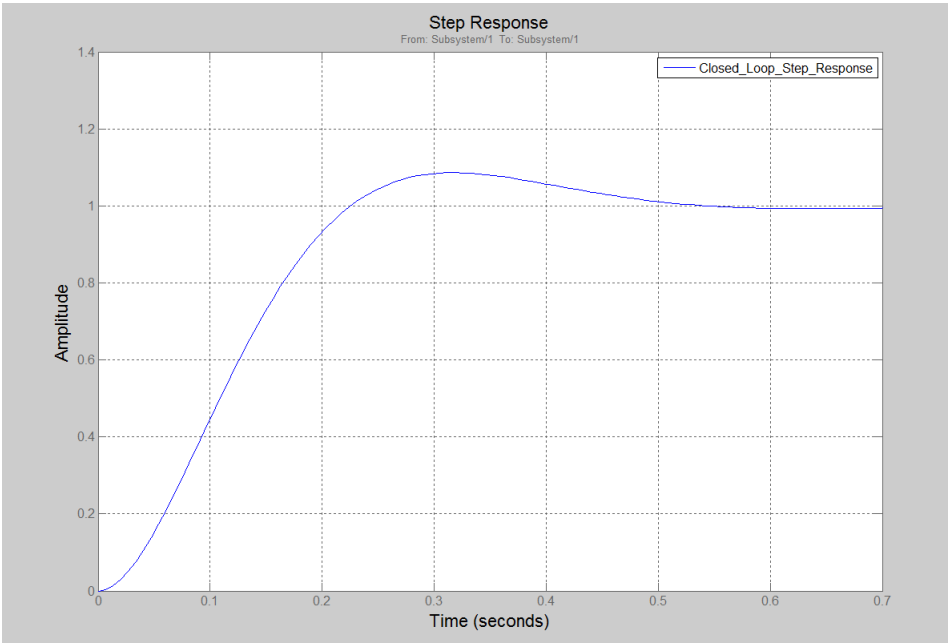


Figure 3.34: Step Response of the Closed Loop(Inner Loop)

According to the Figure 3.34, the settling time for the inner loop is around 0.5 seconds for a step input. System has an overshoot around %10 .

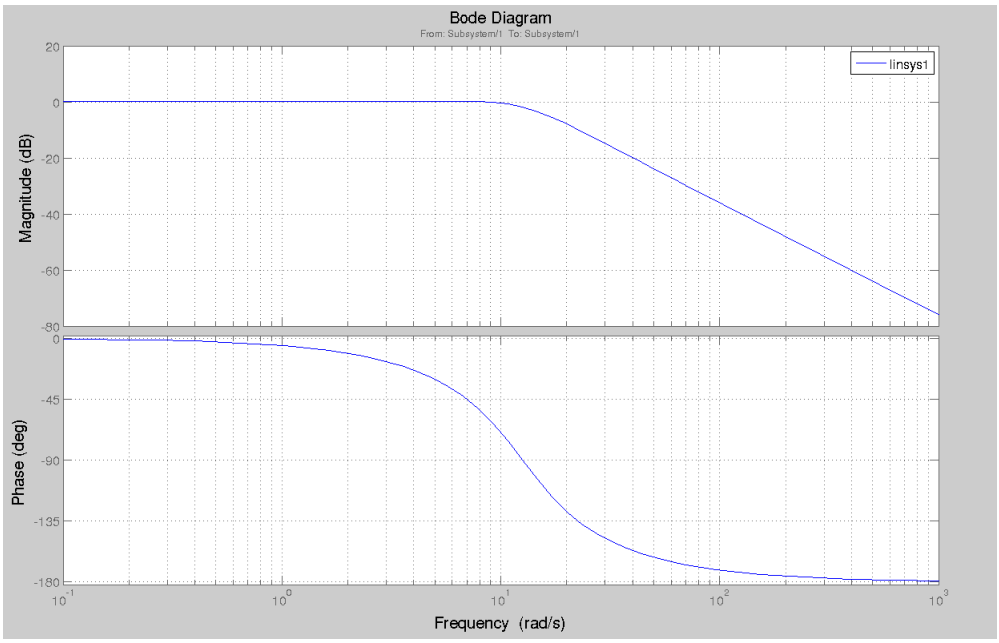


Figure 3.35: Closed Loop Bode Plot(Inner Loop)

Figure 3.35 shows the bode plot for the closed loop of the velocity controller. According to this figure inner loop which control the desired velocity setpoint has a bandwidth nearly 10 rad/sec.

CHAPTER 4

SIMULATION ENVIRONMENT & RESULTS

In this section, the simulation environment is illustrated and the results are analyzed and discussed in detail for the local positioning system design and formation control system design. Three different methods of formation control system which are presented in Section 3 are evaluated and compared with each other. Simulations are handled in the environment of Gazebo simulator with a swarm of 50 agents which includes three different types of robots. We have determined the position beacon ratio as %15, so we have 8 position beacons in our swarm. The main system frequency in which each agent propagates its state vector, calculates artificial forces and makes goal state assignments is determined as 2Hz.

4.1 Simulation Environment

Block diagram of the simulation environment is illustrated in Figure 4.1.

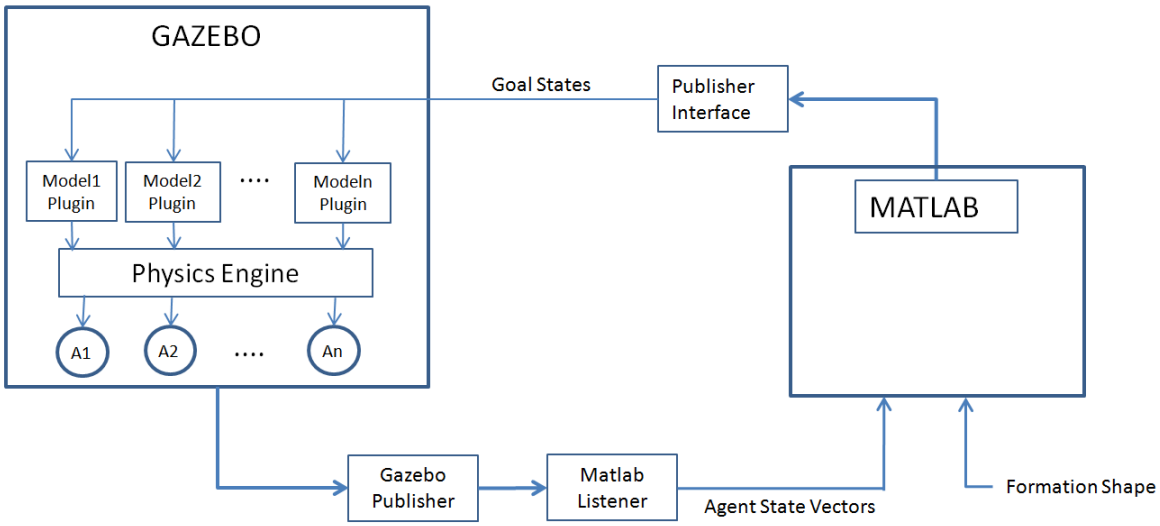


Figure 4.1: Simulation Environment

Gazebo is an open source multi robot simulator developed by Open Source Robotics Foundation(OSRF). It has multiple physics engines; Open Dynamics Engine(ODE), Bullet, Dynamic Animation and Robotics Toolkit(DART) and Simbody. It can execute simulations with multiple agents in an environment that is fully created by the user. The dynamics and the physical properties of the robots are determined by the user. It can be integrated with Robot Operating System(ROS). We have used this simulator to run the dynamics of the agents with the help of ODE physics engine. We have created plugins dedicated on each agent to execute the algorithms in a decentralized topology as discussed in Section 1.3.3. Agents use these plugins to listen goal states from Matlab environment, execute their decision process algorithms and calculate their artificial forces. These plugins also execute controller algorithms to make the agents reached to the goal states they have assigned.

Shape partitioning algorithms are executed in Matlab environment and goal states are published over a network socket. A plugin working in Gazebo environment, listens the packets sent by Matlab and parse the data for each agent type. The state vectors of the agents are published with a plugin from the Gazebo simulator and they are

used to visualize the workpace in a Matlab plot. We have designed the workspace as a rough 3D territory and each agent have different interactions with the environment with different mass and friction characteristics. This workspace is defined with a square area with boundaries placed at +/-50 meters in x and y coordinates. We have assumed that the communication range of an agent is 25 meters. To simulate the radio link between any two agent, we use a line segment drawn between the centers of these agents' coverage circles. It is assumed that these two agents have a direct radio link (i.e. they are direct neighbors in the mesh network) if the p_2 norm of this vector is smaller than the communication range and this line segment doesn't intersect any workspace obstacles.

Figure 4.2 illustrates the Gazebo simulation environment.

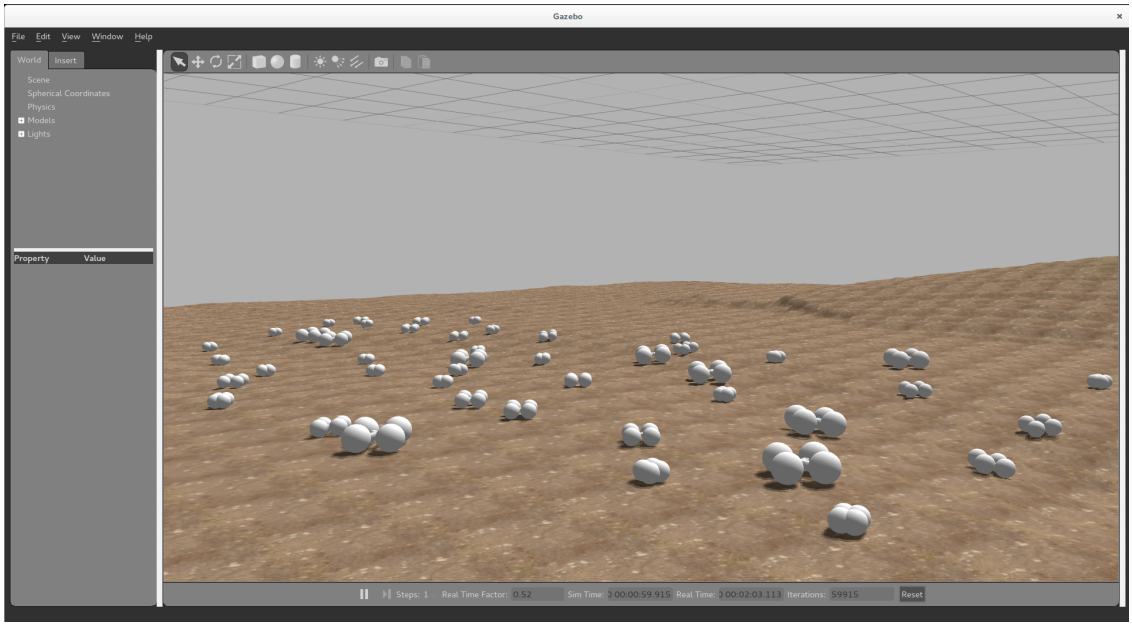


Figure 4.2: Simulation Environment in Gazebo

In this simulation environment, each agent is designed with collision surfaces composed by four spheres which are assumed to have point interaction with the environment. Agents have cylindrical links binding these spherical contact points. The mass and friction coefficients for different types of agents are given in Table 4.1. Here there are two different types of frictions, μ_1 represents the dry friction coefficient and the μ_2 is the fluid friction coefficient.

Table 4.1: Parameters for Different Types of Agents

Agent Type	**Mass[kg]**	μ_1	μ_2	**Coverage Circle Radius[m]**
Agent 1	0.8	0.21	0.13	0.15
Agent 2	1	0.18	0.18	0.3
Agent 3	1.5	0.21	0.13	0.5

As illustrated in Section 3.2.4, full state feedback gains for the agents are calculated with the help of LQR methodology by using the linear system definitions of different types of agents. Calculated feedback gains are given in Table 4.2.

Table 4.2: State Feedback Gains for Different Types of Agents

Agent Type	K_e	K_v
Agent 1	237	96
Agent 2	263	112
Agent 3	297	121

Figure 4.3 illustrates the visualization of the workspace in the Matlab environment. Red circles which are the same size with coverage circles defined in Section 3.2.1.1, are representing the different types of agents in the environment. The shape defined with blue line is the desired formation shape and the obstacles are represented with green lines in the environment.

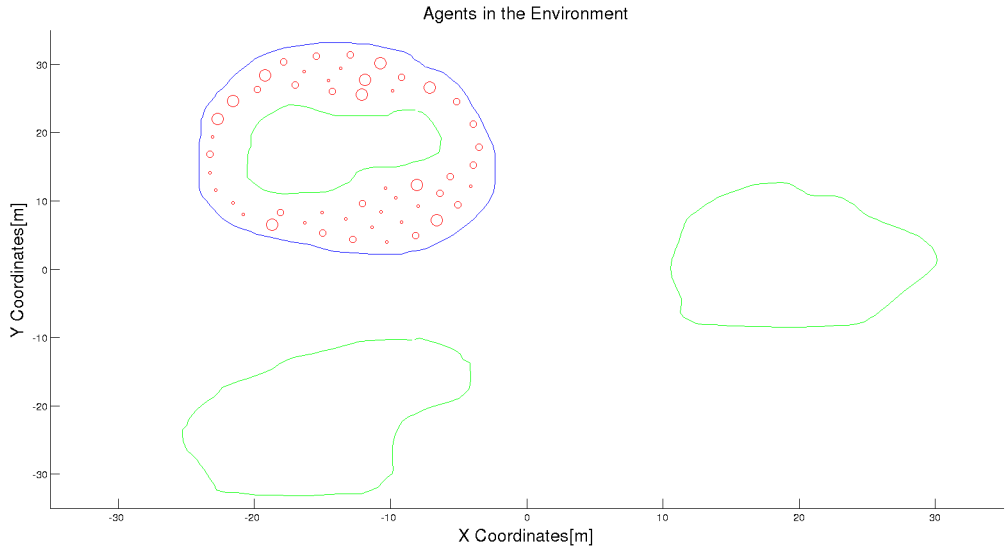


Figure 4.3: Visualization of Workspace in Matlab

4.2 Local Positioning System

In the local positioning system, the main aim is to design an architecture in which every second type agent can localize itself with the help of the position beacons. In this architecture, second type agents need external position measurements for their state estimator systems to prevent a potential drift in their position data. In our implementation, the state propagation frequency is determined as 2Hz which is equal to the main system frequency since the new position datas for the formation control system are required to be determined with this rate. As discussed in Section 3.1, a localization timer is proposed to provide the required minimum time to the DSDV algorithm and trilateration process.

We are expecting that the position data in the state vector will be drifting with the increasing time because of the errors and noise in inertial measurements unless they are not corrected with the external measurements [34]. So it is possible to determine a minimum execution period of this trilateration process to satisfy a maximum error in the position data. For this purpose we have executed Monte Carlo simulations to observe the drift in the position data with state propagation process. In Figure 4.4, 4.5 and 4.6, the simulation outputs with different periods of localization timer are illustrated.

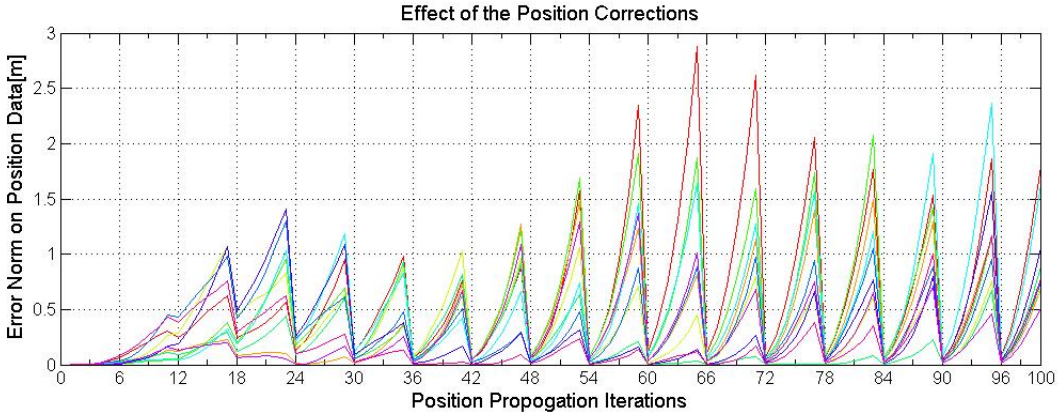


Figure 4.4: Total Error with Localization Timer Period of 3 Seconds

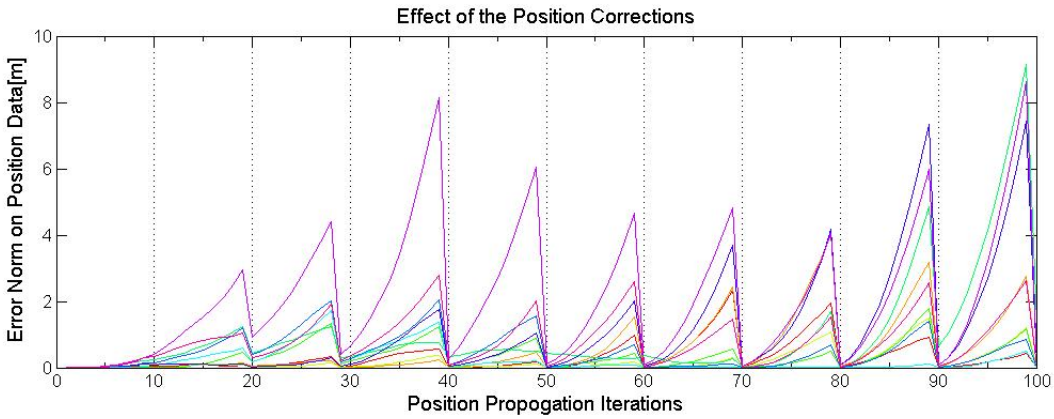


Figure 4.5: Total Error with Localization Timer Period of 5 Seconds

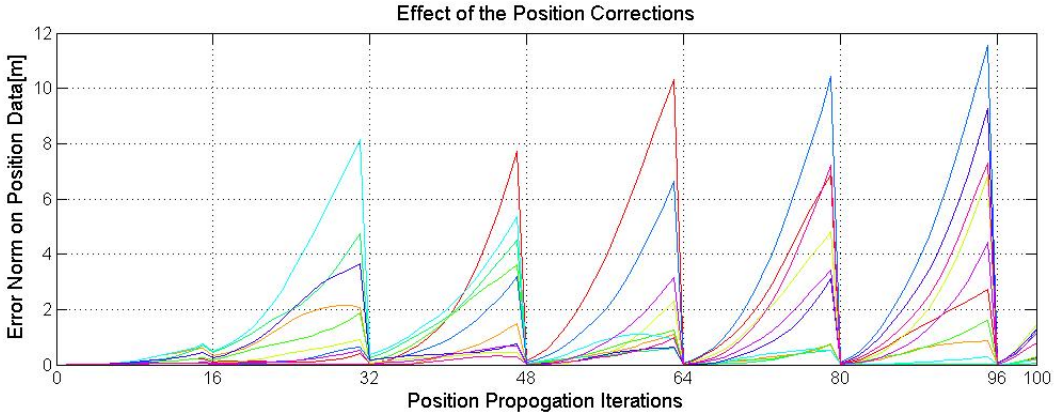


Figure 4.6: Total Error with Localization Timer Period of 8 Seconds

According to these figures, it is clear that the total error norm on position datas are decreased dramatically with the localization period which equals to 6 iterations for 3 seconds period, 10 iterations for 5 seconds and 16 iterations for 8 seconds period. The peak values on the error norms are always observed at the iterations just before the localization process. It can be concluded that the peak values on the error norms are greatly related with the localization period and they have greater values with the increasing number of this period. The maximum error norm is below 3 meters with localization period of 3 seconds, below 10 meters with localization period of 5 seconds and below 12 meters with localization period of 8 seconds. In this work, it is assumed that the error norm of 3 meters is the maximum tolerable value for the formation control system to satisfy a successful collision avoidance for the agents since we have a coverage circle radius of 0.5m for the biggest agent in the swarm.

Thus, the localization period for the trilateration process is determined as 3 seconds in simulations.

The performance of the LPS design is tested in simulations with different conditions in which swarm is propagating to a desired formation shape or agents are keeping their positions in dynamically changing formation shapes. In both cases, it is observed that the position datas of the agents are drifted with an increasing error between two sequential localization process as we have discussed previously. A sample case for this situation is illustrated in Figure 4.7 and Figure 4.8. In these figures, red circles are representing the estimated positions of the agents where blue circles are representing the real positions.

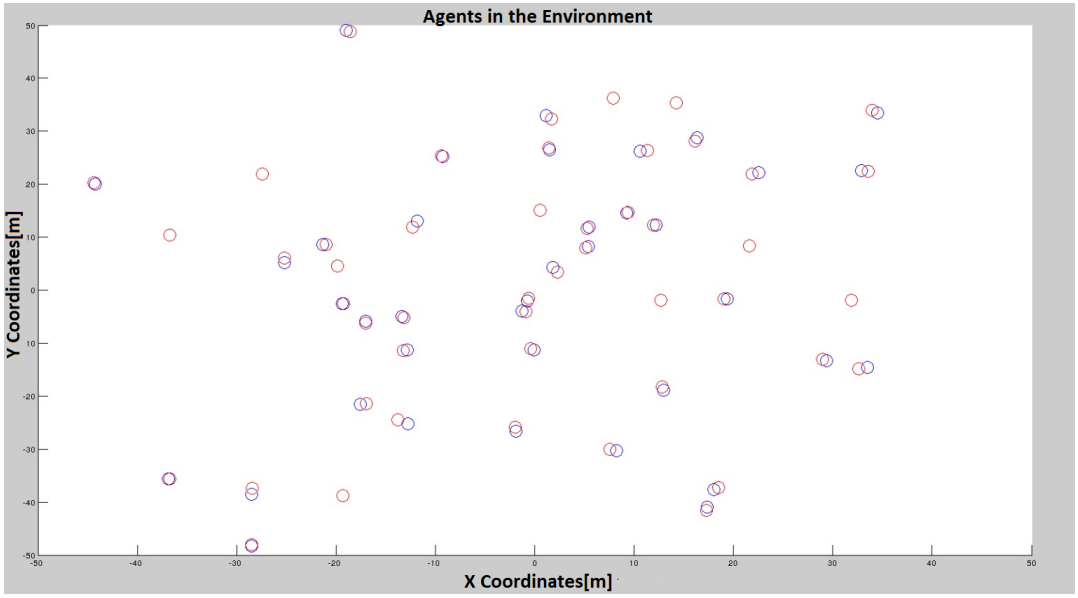


Figure 4.7: Positions of the Agents Before Localization Process

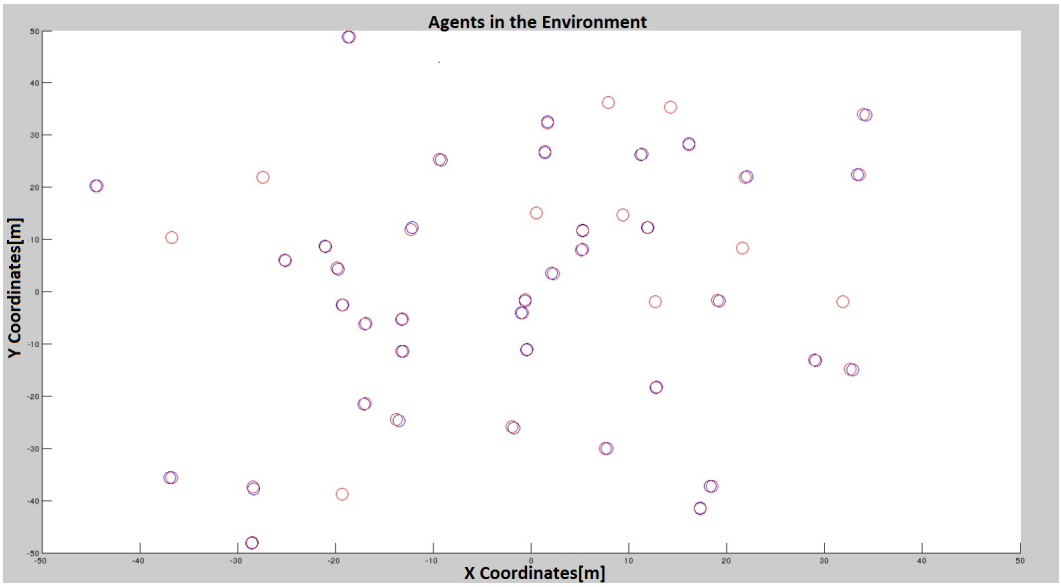


Figure 4.8: Positions of the Agents After Localization Process

In Figure 4.7, blue and red circles do not coincide with each other. This shows us, there are errors on the estimated positions which are represented with red circles before the localization process. Figure 4.8 shows the workspace just after a localization process and it is possible to see that blue and red circles are getting close to each other with this process. This concludes that the error on the estimated position datas are reduced with the help of this process.

Handling procedures of the lost agents are presented in Section 3.1.3. Agents which do not have at least three position beacons which are direct neighbors around itselves will get into the 'Lost' mode, and if they miss three localization process they will get into 'Come Home' mode. When an agent is in 'Come Home' mode, it will try to reach to the center of formation shape to increase the possibility of meeting some position beacons to localize itself. A simulation result in which two agents do not have three position beacons as direct neighbors around themselves is illustrated in following figures. They get into 'Lost' mode first, when they cannot find three position beacons as direct neighbors. After they miss three localization process, they get into 'Come Home' mode in which they are trying to reach to the center of formation shape.

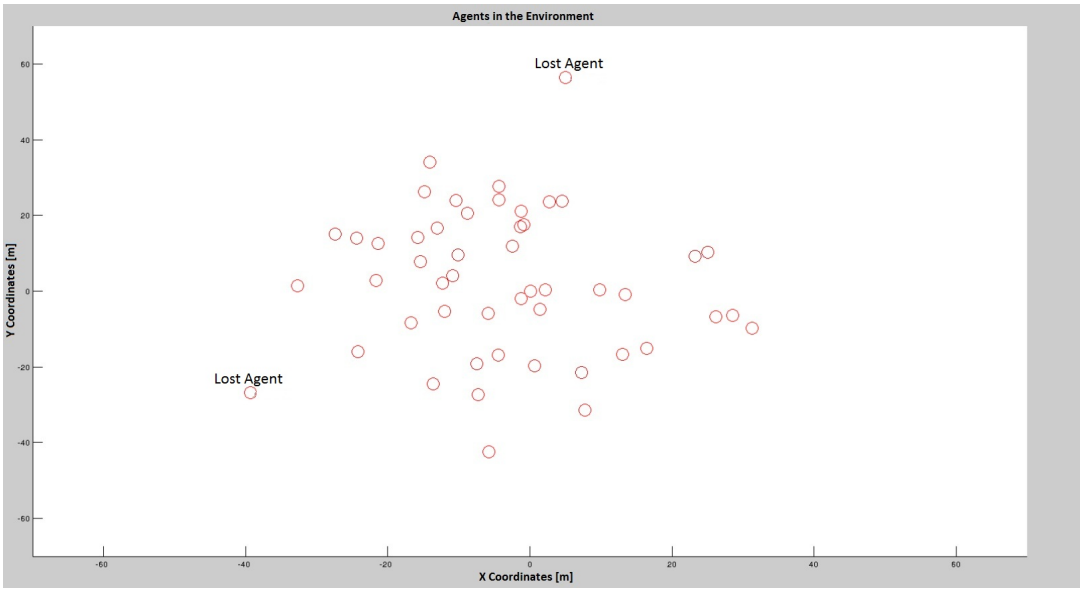


Figure 4.9: Agents will be in 'Lost' Mode When They Do Not Have 3 Neighbors

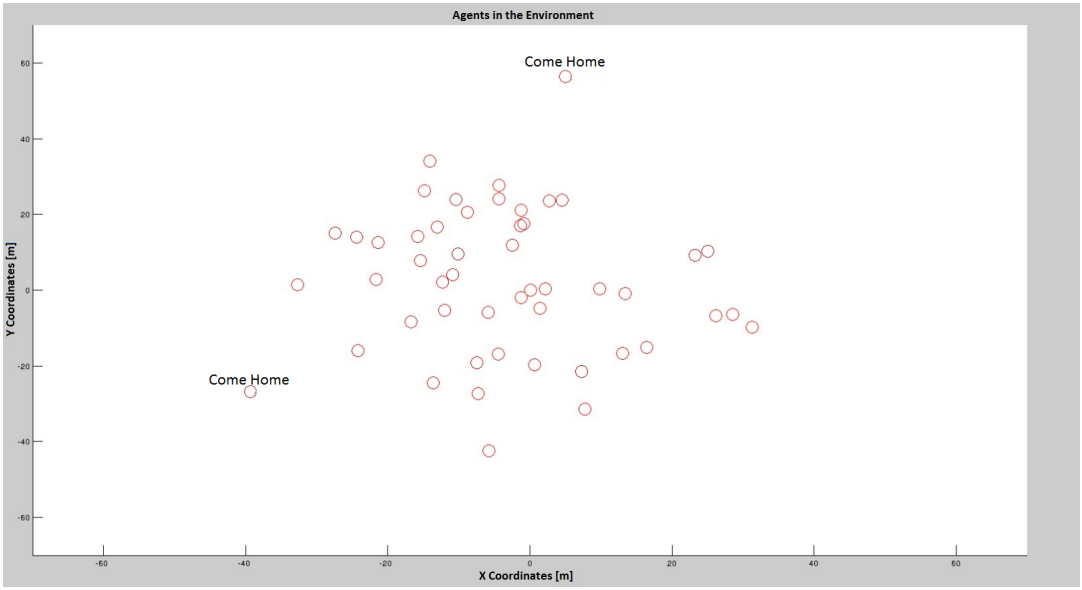


Figure 4.10: Agents will be in 'Come Home' Mode After 3 Localization Period

4.3 FORMATION CONTROL SYSTEM

In this section, three different methods of which the details are presented in Section 3.2 are evaluated according to their settling times, mesh qualities and total displacements.

4.3.1 Mesh Qualities

Mesh quality is a measure of how the agents homogeneously distributed while covering the desired formation shape. Basically, two different types of quality measurements defined in Section 3.2.3 are used to evaluate the performance of the formation control methods, topological mesh irregularity ε_t and geometrical mesh irregularity ε_g . Monte Carlo simulations with 1000 iterations are handled for the same formation shapes with different initial conditions of the agents in the environment. Sample outputs for three different types of formation control algorithms are illustrated in the following figures. In these figures, red circles are representing the coverage circles of three different type of agents. Mesh irregularities are calculated by constructing the Voronoi diagrams with nodes located at the centers of these coverage circles.

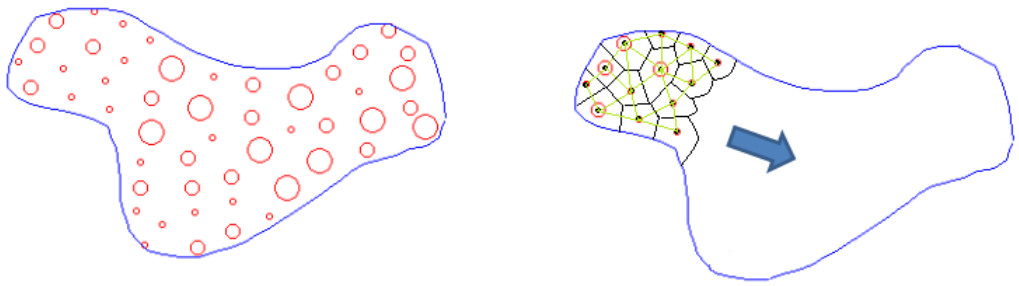


Figure 4.11: Shape 1 with Artificial Forces Methods:$\varepsilon_t = 2.1$ and $\varepsilon_g = 0.32$

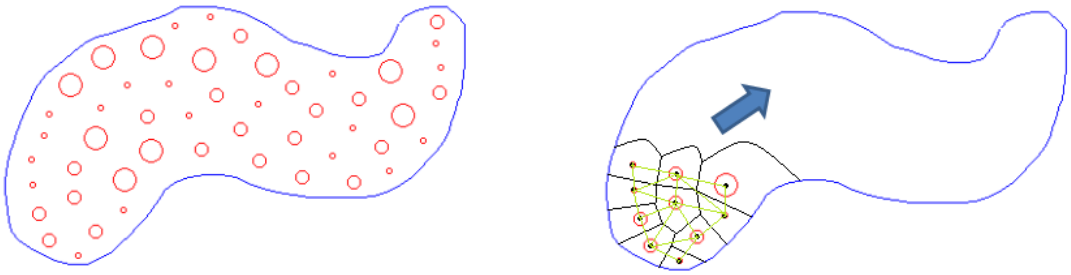


Figure 4.12: Shape 2 with Artificial Forces Methods:$\varepsilon_t = 2.6$ and $\varepsilon_g = 0.4$

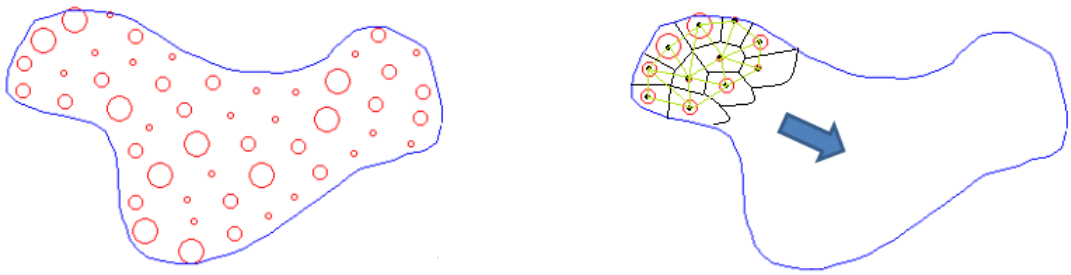


Figure 4.13: Shape 1 with Bubble Packing Method:$\varepsilon_t = 2.1$ and $\varepsilon_g = 0.24$

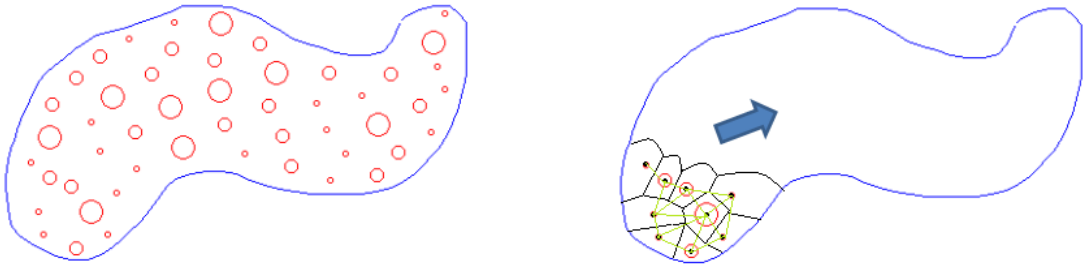


Figure 4.14: Shape 2 with Bubble Packing Method:$\varepsilon_t = 2.3$ and $\varepsilon_g = 0.28$

Figure 4.15: Shape 1 with Randomized Fractals Method:$\varepsilon_t = 2.6$ and $\varepsilon_g = 0.63$

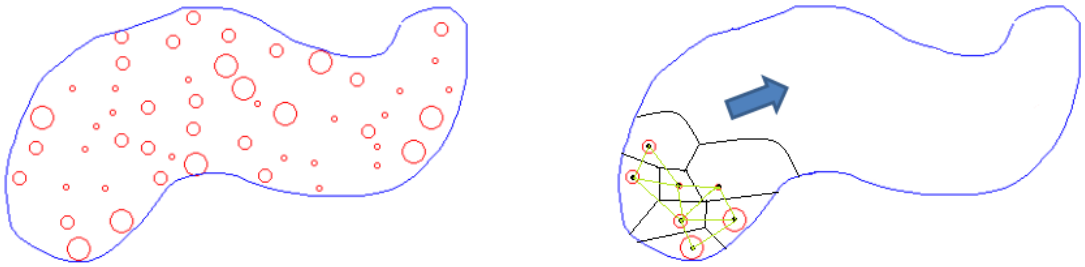


Figure 4.16: Shape 2 with Randomized Fractal Method:$\varepsilon_t = 3.1$ and $\varepsilon_g = 0.61$

It can be concluded that Randomized Fractals method has the worst mesh performance. It is because of the randomized assignment of goal states into the formation shape in this methodology. Bubble Packing and Artificial Forces methods give similar results since they have an analogy in their approaches of implementing inter member/bubble forces which makes the agents/bubbles distributed in the formation shape more homogeneously. Artificial force method applies this intermember forces directly to the agents in formation control while Bubble Packing method use this force on artificial bubbles to partition the shape into goal states. Monte Carlo simulations with 1000 iterations for both formation shapes with different initial conditions are handled and the results are illustrated in Figure 4.17; 4.18; 4.19; 4.20 . Randomized Fractals method have greater mean values for both topological and geometrical mesh irregularities as expected. Bubble Packing and Artificial Forces methods have similar performances on mesh quality.

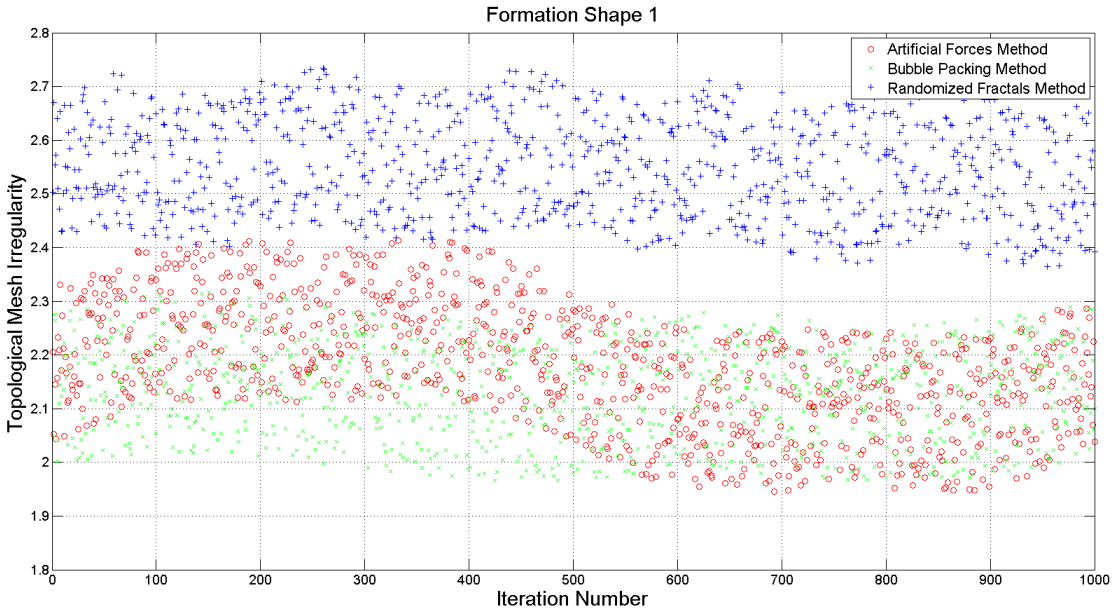


Figure 4.17: Formation Shape 1 Topological Mesh Irregularities

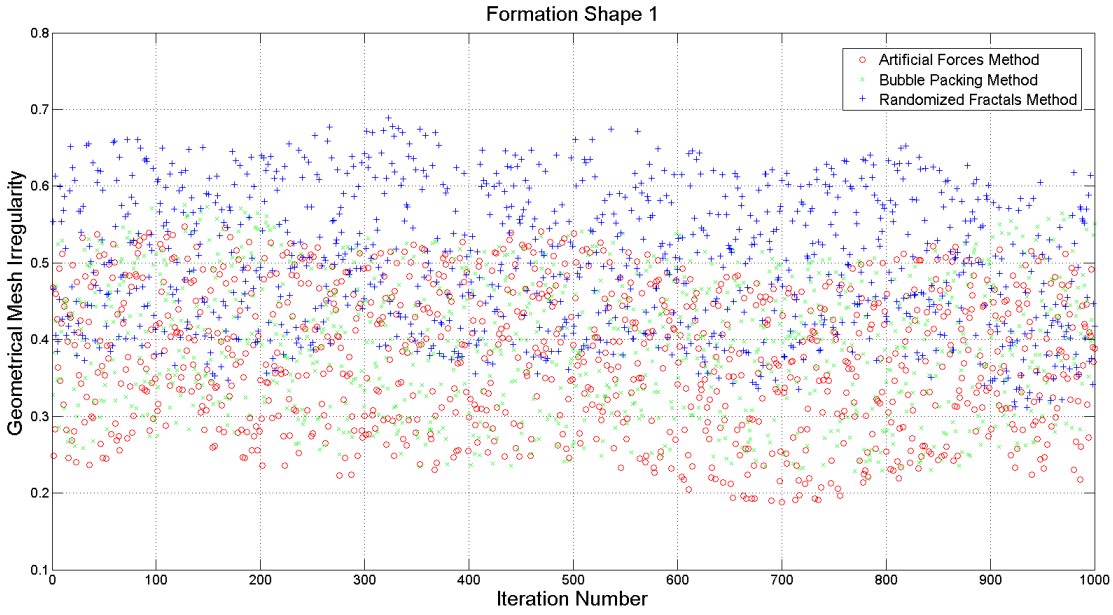


Figure 4.18: Formation Shape 1 Geometrical Mesh Irregularities

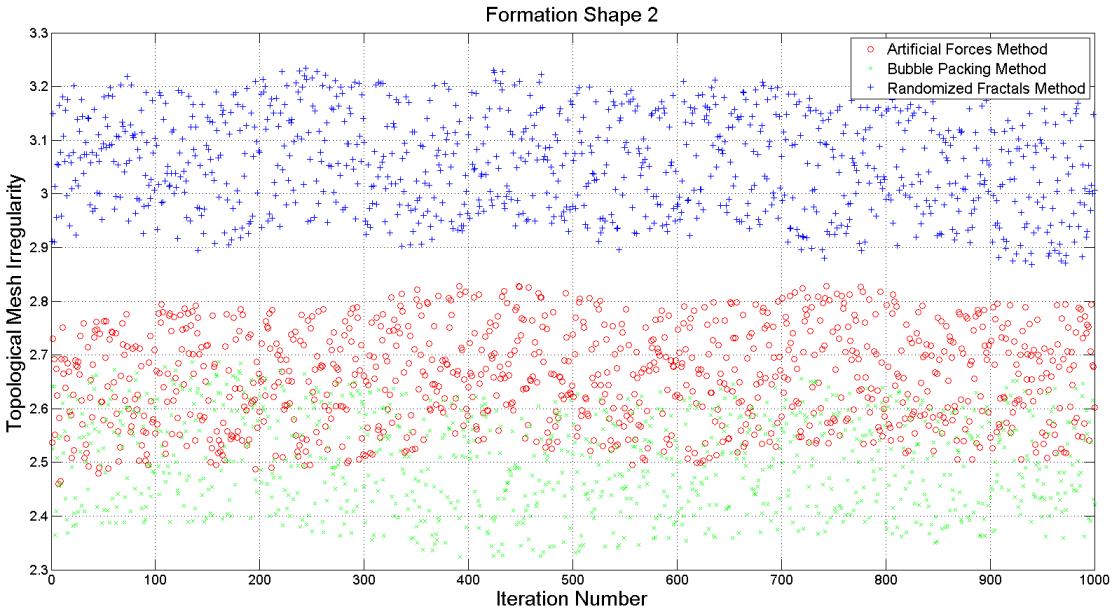


Figure 4.19: Formation Shape 2 Topological Mesh Irregularities

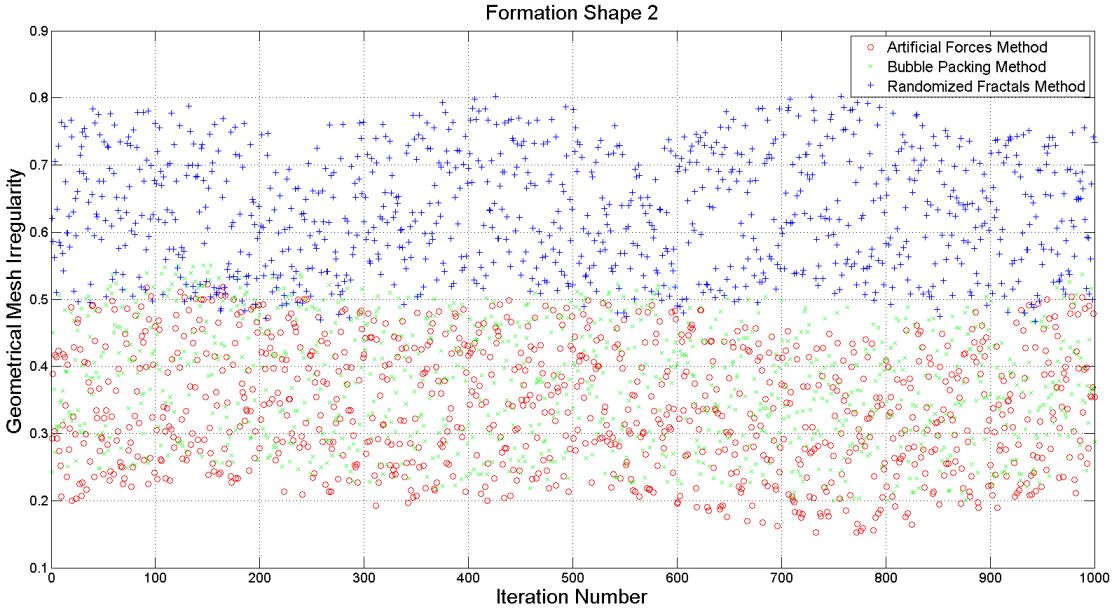


Figure 4.20: Formation Shape 2 Geometrical Mesh Irregularities

4.3.2 Total Displacement of the Agents

In this project, we aim to minimize the overall displacement of the agents in the swarm while getting the desired formation shape. Here the main approach is to reduce down the expected energy consumption of the swarm by decreasing the required displacements. To compare the total displacement of the swarm while getting the desired shape for three different methods, we have done Monte Carlo simulations with 1000 iterations. These simulations are handled for the same formation shapes with different initial conditions of the agents in the environment. Trajectories of the agents are recorded from the initial positions to the goal states for three different types of formation control systems. Sample outputs for three different types of formation control algorithms are illustrated in the following figures.

4.3.2.1 Shape 1

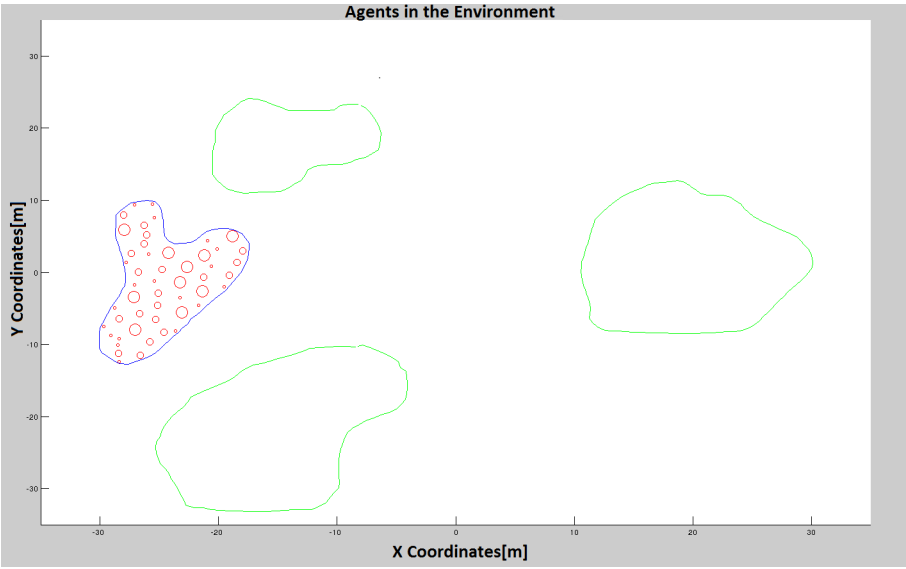


Figure 4.21: Formation Shape 1 in MATLAB environment

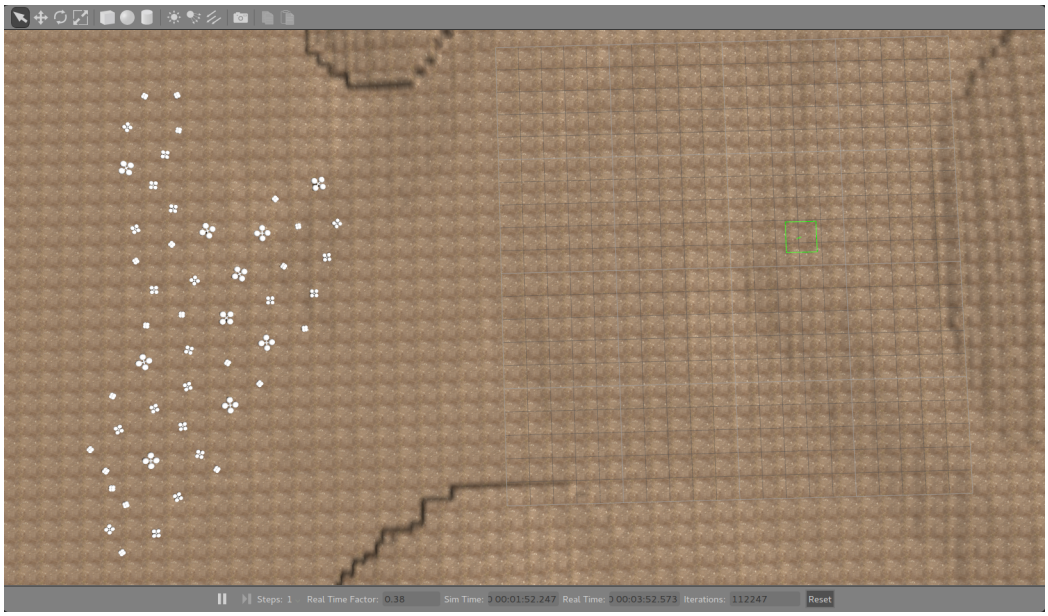


Figure 4.22: Formation Shape 1 in Gazebo environment

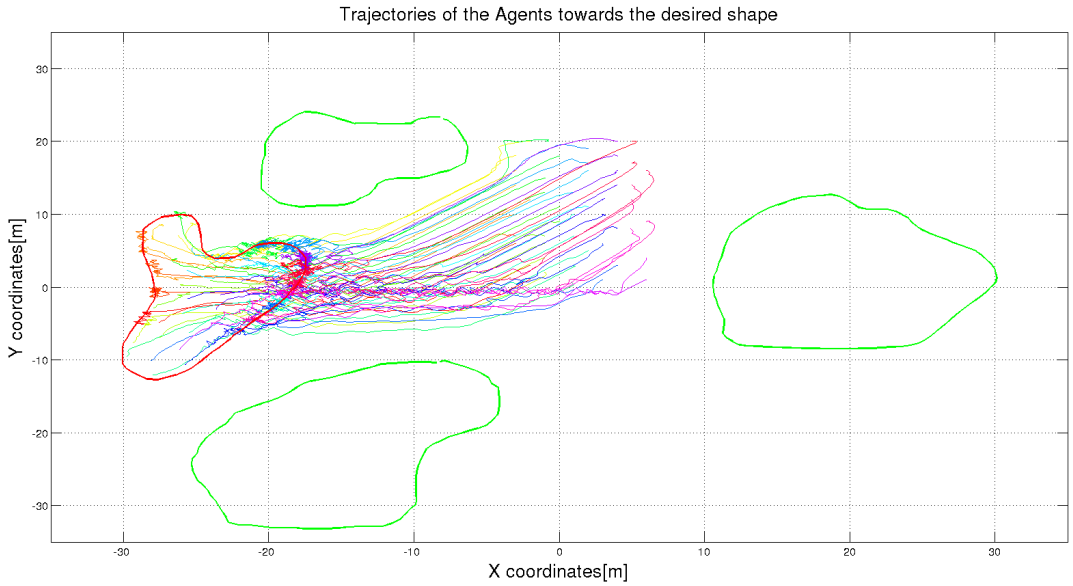


Figure 4.23: Artificial Forces Method Trajectories for Shape 1

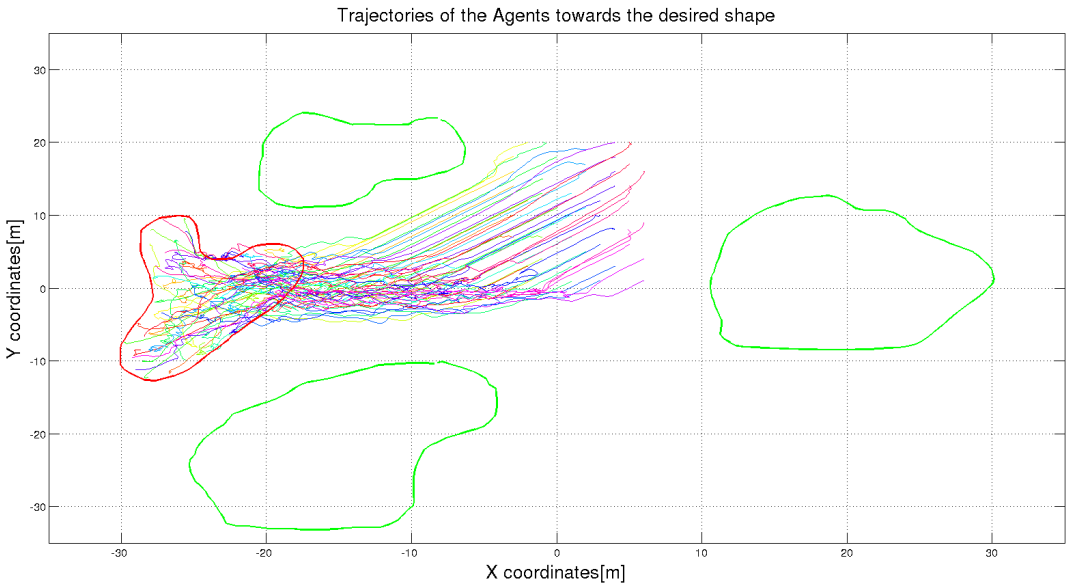


Figure 4.24: Bubble Packing Method Trajectories for Shape 1

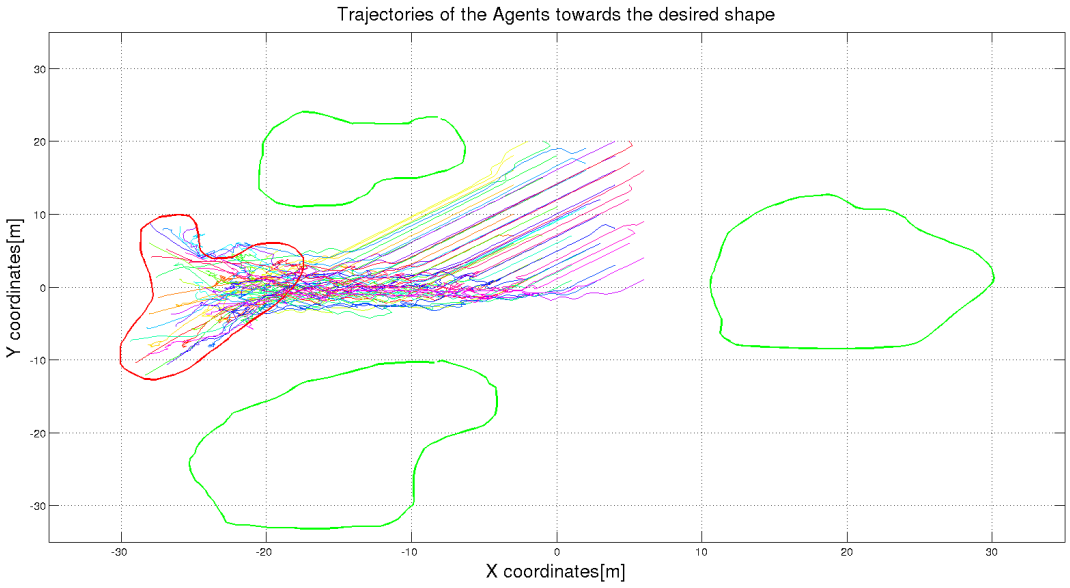


Figure 4.25: Randomized Fractals Method Trajectories for Shape 1

It can be seen that the trajectories of the agents are more chaotic and complex in the Artificial Forces method according to the other two solutions. Main cause for this situation, the agents are under the effect of a total force which is instantly changing both amplitude and direction with the local interactions with the environment in Ar-

tificial Forces method. The only force component which provides the distribution of the agents in the formation shape homogeneously is the intermember forces which is dynamically changing so much with the local instant neighbors of the agents in the environment. On the other hand, Bubble Packing and Randomized Fractal methods implement an algorithm in which every agent is directed to a goal state in which the total displacements in the environment is minimized. This approach prevents the chaotic appearance of the trajectories and minimizes the displacements of the agents. Because agent are trying to reach their goal states directly within these two methods.

To compare the total displacements, we have done Monte Carlo simulations with 1000 iterations and the results are presented in Figure 4.26. According to this figure, Artificial Forces method has a higher mean value of total travelled distance while achieving the same formation shape with same initial conditions. Performances of Bubble Packing and Randomized Fractals method seem similar, because the only difference between these two methods is their shape partitioning approaches. The assignment procedure of the agents to these goal states by minimizing the total displacement is implemented identically.

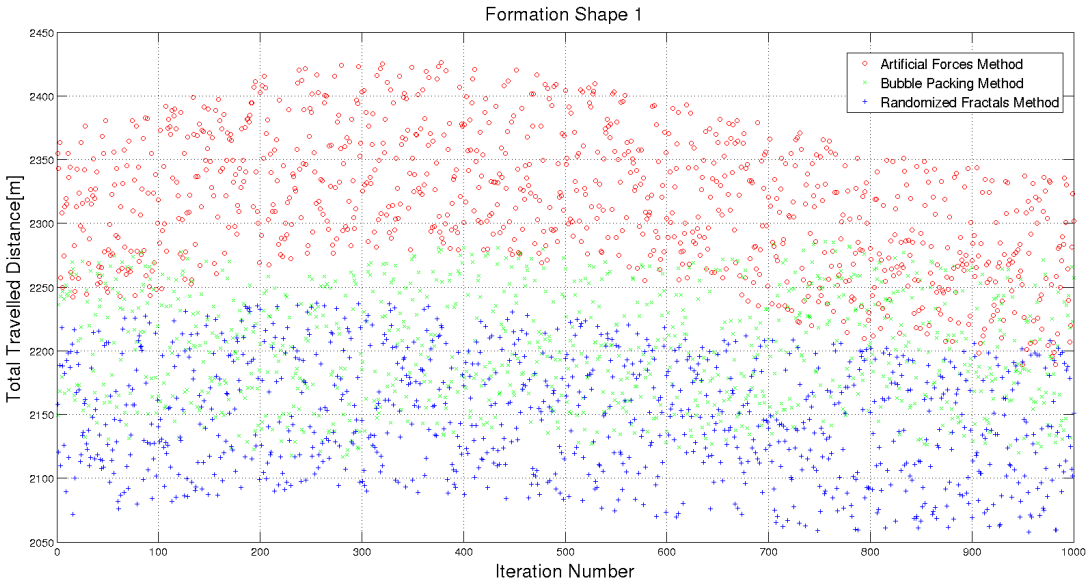


Figure 4.26: Total Travelled Distances for Shape 1

4.3.2.2 Shape 2

Second formation shape gives similar results with three different formation control methods, because of the reasons discussed for the Formation Shape 1. Artifical Forces method has the worst performance on total travelled distance metric.

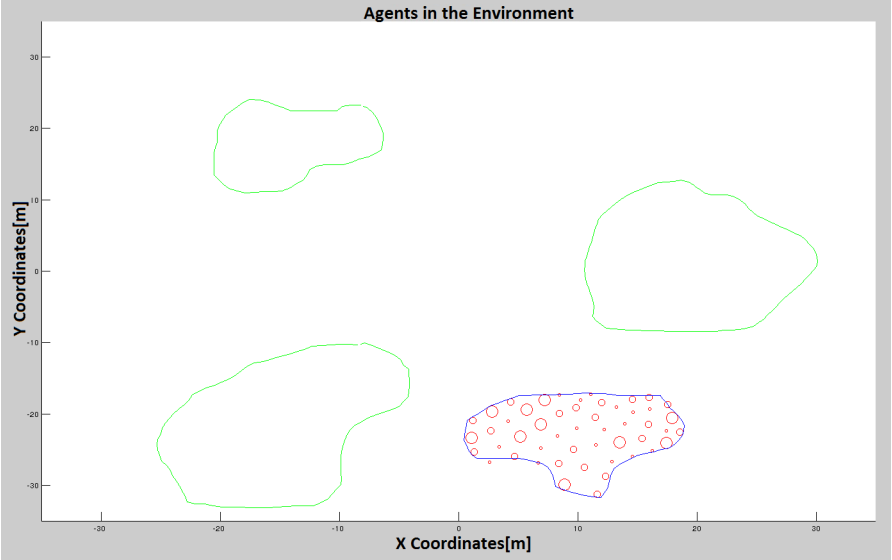


Figure 4.27: Formation Shape 2 in MATLAB environment

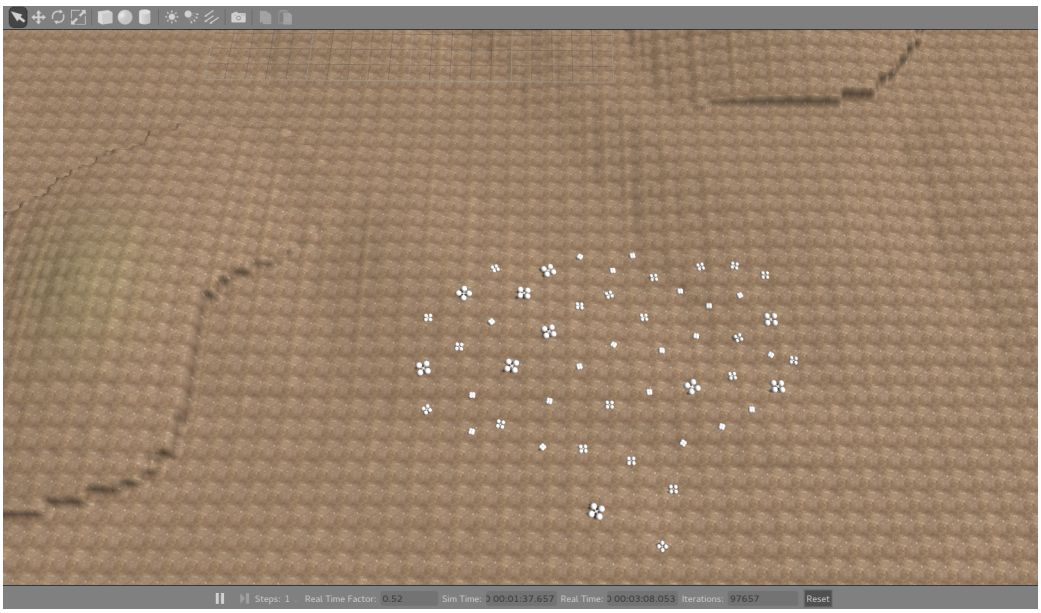


Figure 4.28: Formation Shape 2 in Gazebo environment

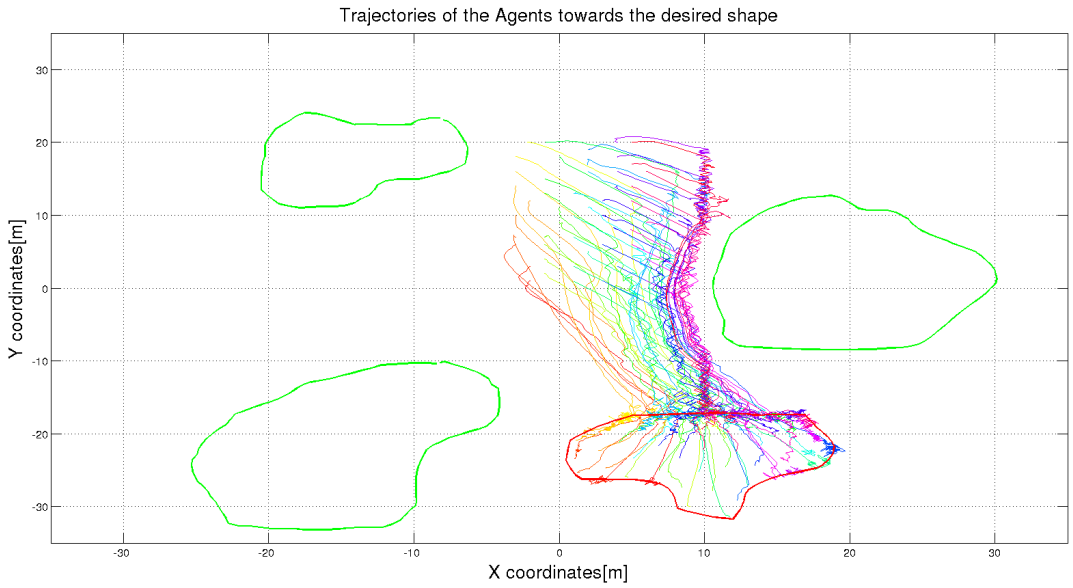


Figure 4.29: Artificial Forces Method Trajectories for Shape 2

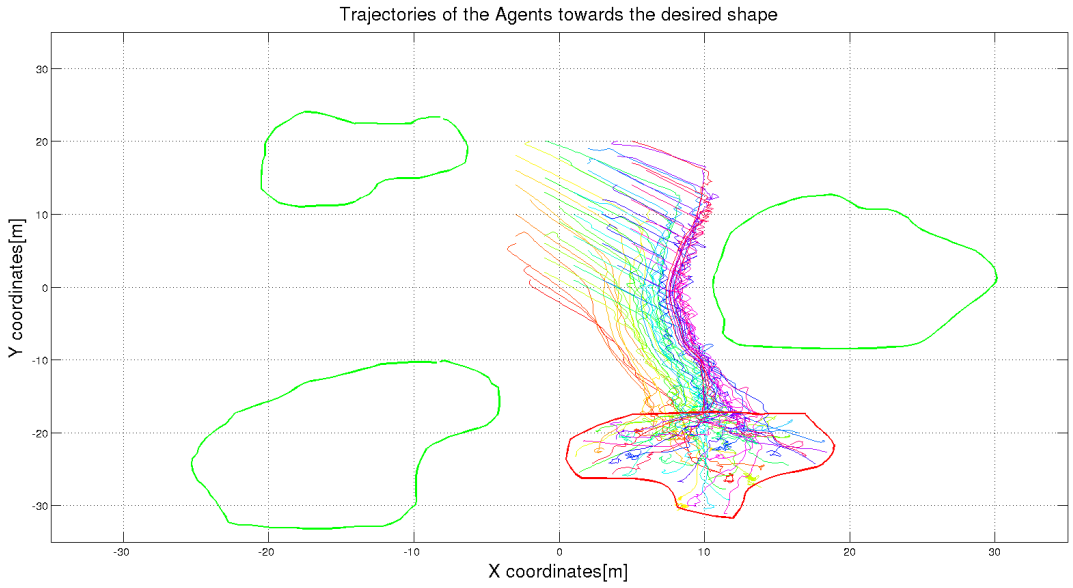


Figure 4.30: Bubble Packing Method Trajectories for Shape 2

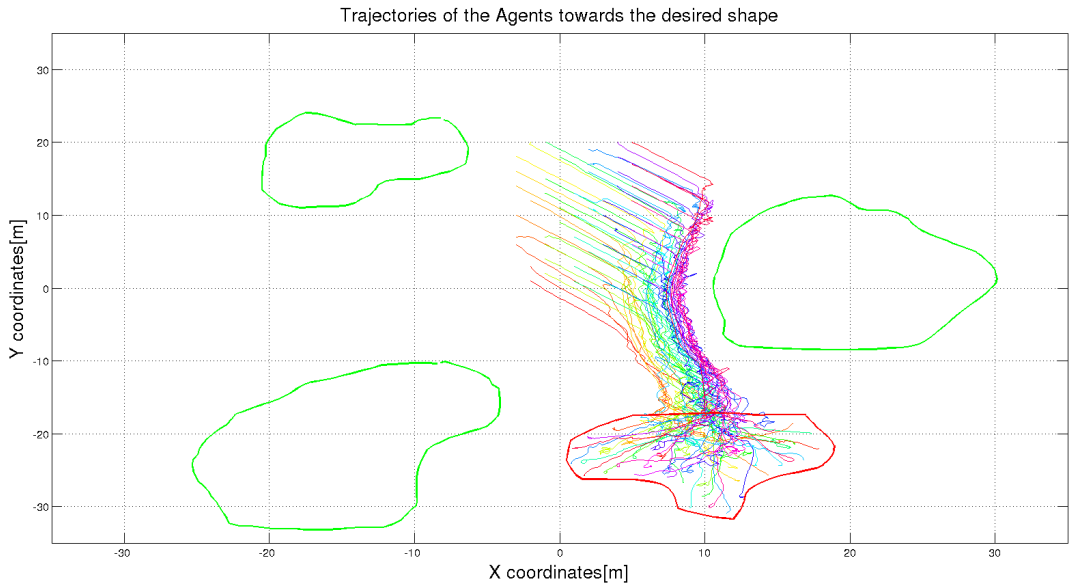


Figure 4.31: Randomized Fractals Method Trajectories for Shape 2

The results of the Monte Carlo simulations with 1000 iterations are illustrated in Figure 4.32 . In this figure, Artificial Forces method has a higher mean value on total travelled distance for the same formation shape with same initial conditions.

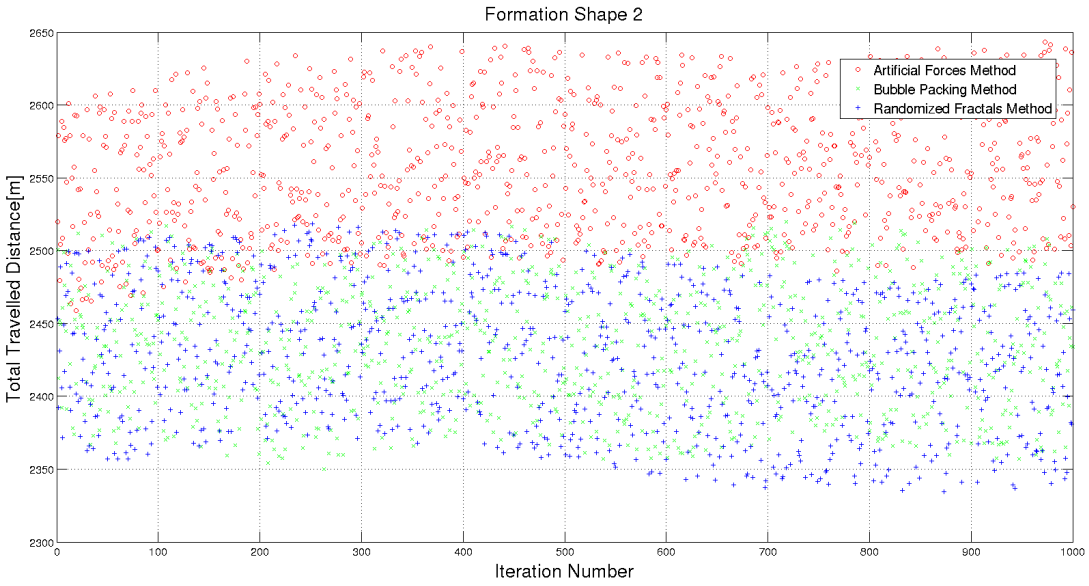


Figure 4.32: Total Travelled Distance for Shape 2

4.3.3 Settling Time

Settling time is defined as delta time "$t_{final} - t_{start}$" ,where t_{start} is the initial time and t_{final} is the time that the all of the agents are inside of the desired formation shape and the norm of the velocity vector for each agent $\|v_i\|$ is

$$\|v_i(t)\| < 0.01\ [m/sec] \quad \forall t > t_{final} \tag{4.1}$$

To compare the settling time performances of three different methods, we have measured the settling times for the simulations held in Section 4.3.2. Artificial Forces method has the worst performance because agents are expected to need more time to reach the steady state under the effect of different artificial force components. Since there are no predetermined goal states for the agents in the desired formation shape, the settling of the agents in random places under the equilibrium of the artificial force components takes more time than the other two formation control methods. The agents have reached the steady state at their goal states with the Bubble Packing and Randomized Fractals methods faster than the Artificial Forces Methods. The results are illustrated in Figure 4.33 and 4.34. According to these figures, Artificial Forces method has a higher mean value on settling time higher than the other two methods.

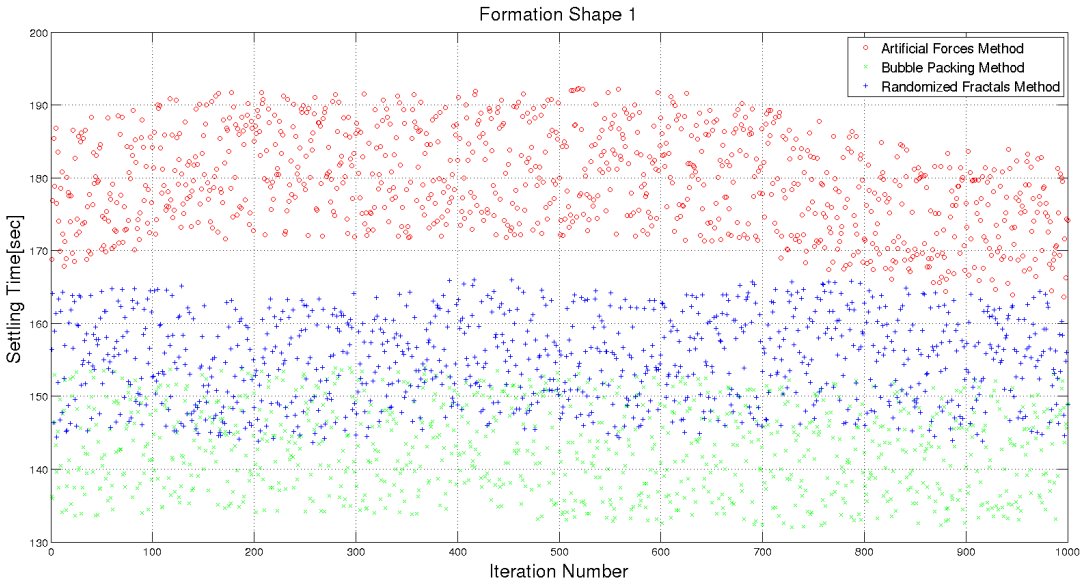


Figure 4.33: Total Settling Time for Shape 1

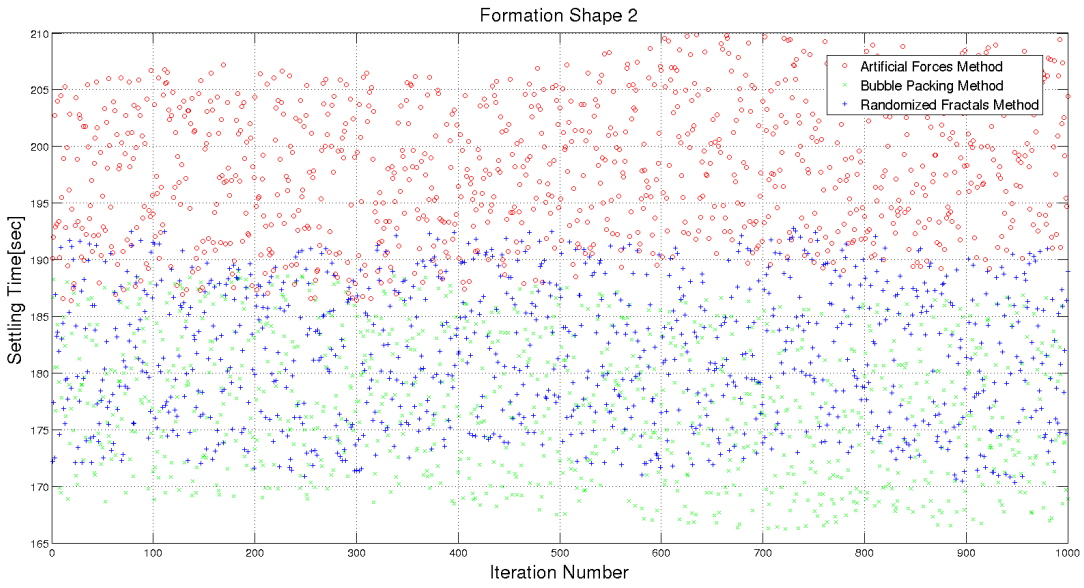


Figure 4.34: Total Settling Time for Shape 2

4.3.4 Dynamically Changing Formations

In dynamic formation control, desired formation shape is changed dynamically in a smooth fashion. Agents have to adapt themselves to this changing formation shape and cover the shape in a continuous manner. Since the formation shape is expected to have smooth changes in local regions, it is aimed to adapt the agents which are positioned at these local regions rather than replacing all agents simultaneously. To minimize the total displacement of the swarm, the general approach is to make the agents which are close to the changing regions take action and replace themselves while other ones which are far away from that region keeps their positions. Randomized fractals method doesn't have a solution for this type of problem since its nature of randomly assignment of agents in a given formation shape. These assignments made by the randomized fractals method will not have continuous transitions on goal states between sequential formation shapes changing with execution period of the algorithm. On the other hand, it is possible to adapt the Bubble Packing algorithm to provide a solution for the dynamically changing formations. Since Bubble Packing algorithm determines goal states in a formation shape by applying interbubble forces, it is possible to keep this shape partitioning process running in background and dy-

namically adapt the goal states to the changing formation shape. The assignments to these goal states with Hungarian algorithm will adapt the swarm to the dynamically changing formation shape. Artificial forces method calculates control inputs for the individuals according to the current environment conditions and the formation shape. There is no need to update the algorithm for dynamically changing formation shapes, it already supports dynamically changing formations.

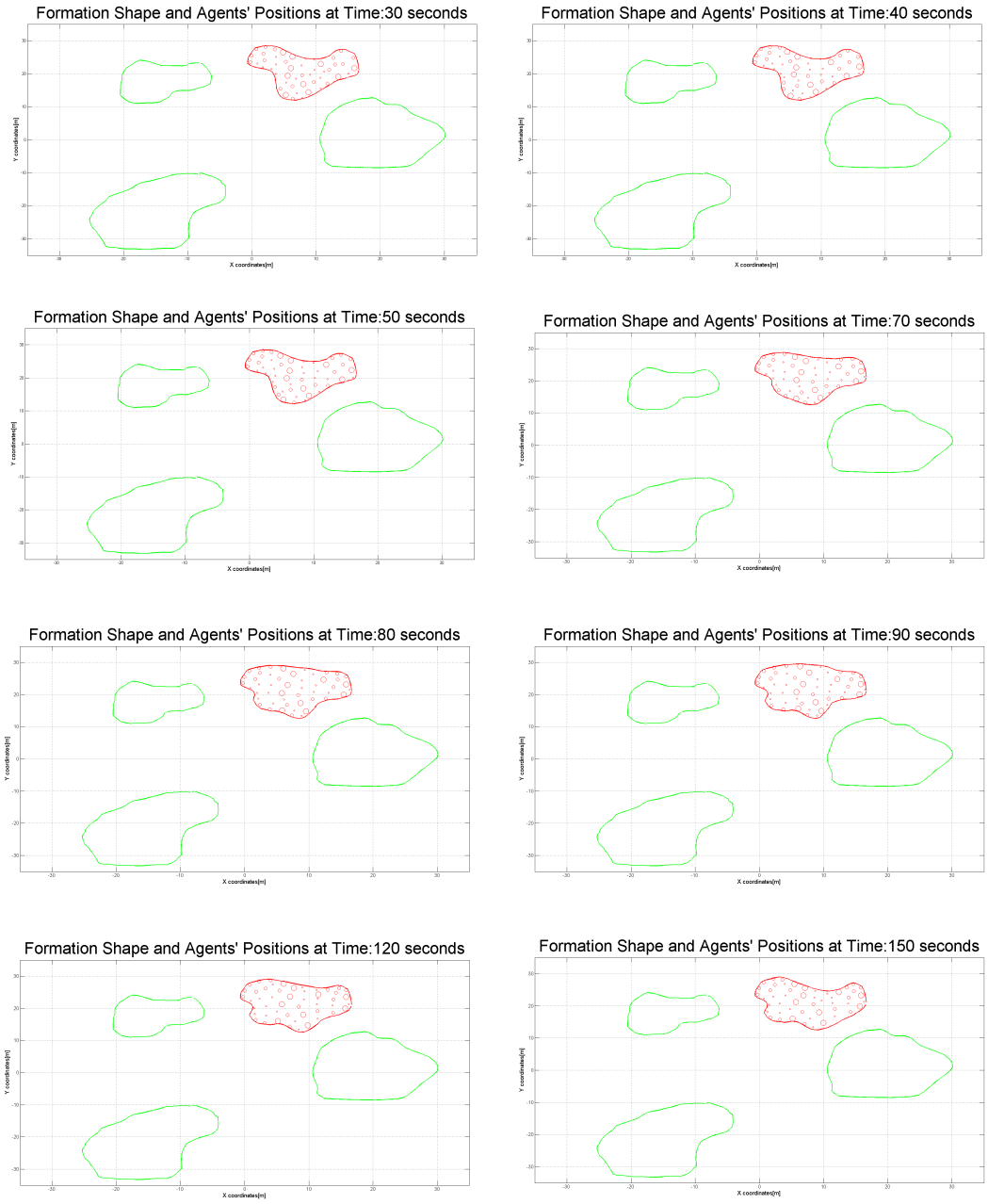


Figure 4.35: Dynamic Formation Control with Bubble Packing Method

Figure 4.35 shows a simulation result for dynamical formation control implemented with Bubble Packing Method. Agents adapt themselves to the dynamically changing formation shape by assigning new positions of the goal states at each iterations. The positions of these goal states are determined with shape partitioning process implemented with Bubble Packing algorithm.

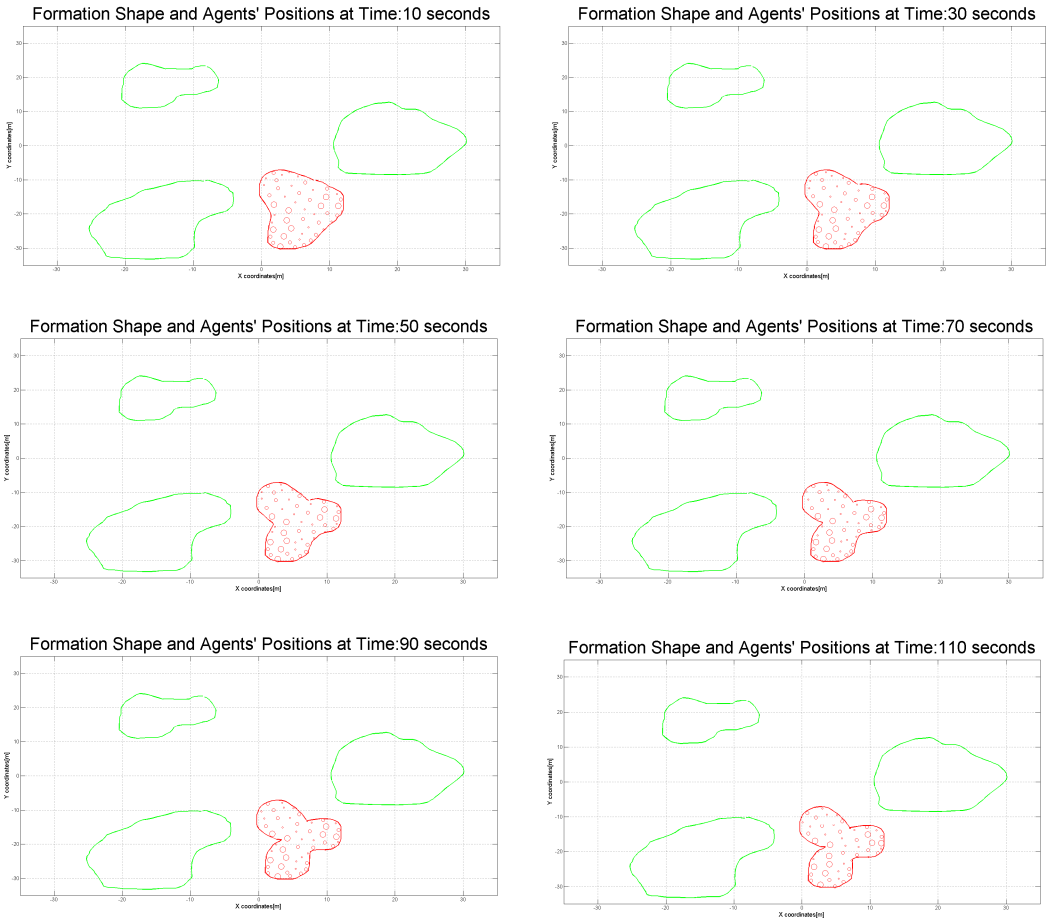


Figure 4.36: Dynamic Formation Control with Artificial Forces Method

Figure 4.36 illustrates a simulation result of Artificial Forces method. In this method agents reposition in the changing formation shape with the help of artificial force components calculated with current formation shape. Since, agents directly adapts themselves to the changing formations shape in Artificial Forces method, it is expected to have a faster response than the Bubble Packing method. In Bubble Packing method, goal states are adapted to the changing formation shape and agents try to reach these goal states in runtime. This kind of approach introduces an additional

latency to the response of the swarm to dynamically changing formation shapes.

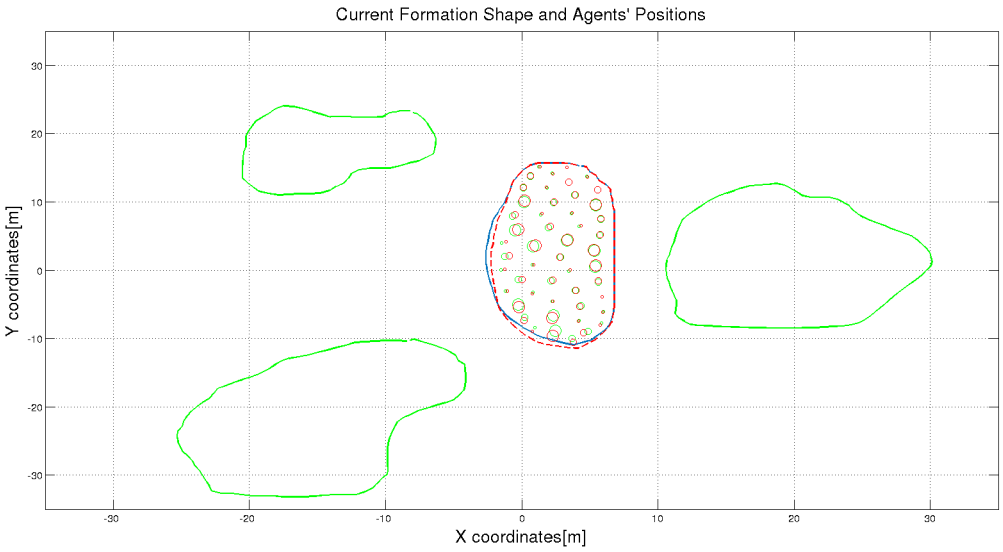


Figure 4.37: Latency in Bubble Packing Method-1

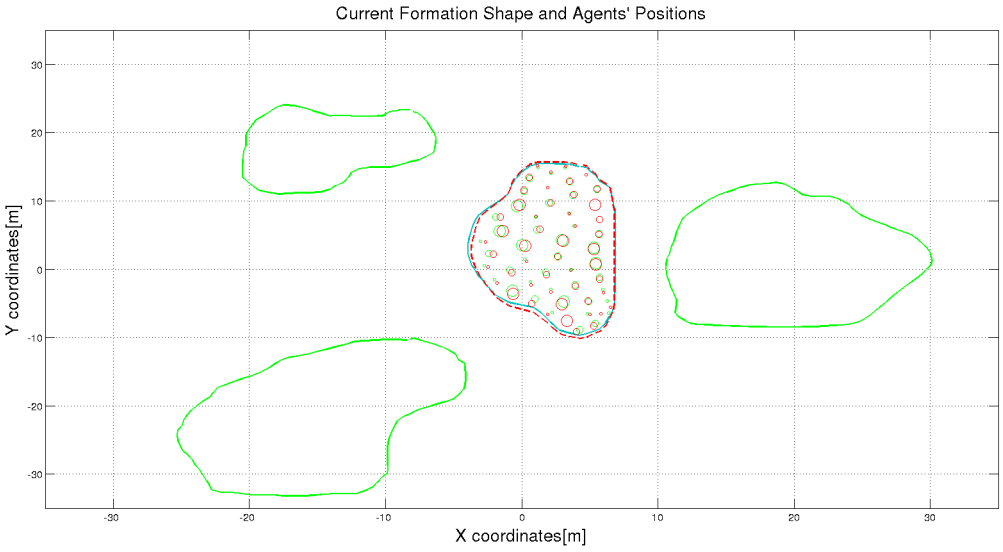


Figure 4.38: Latency in Bubble Packing Method-2

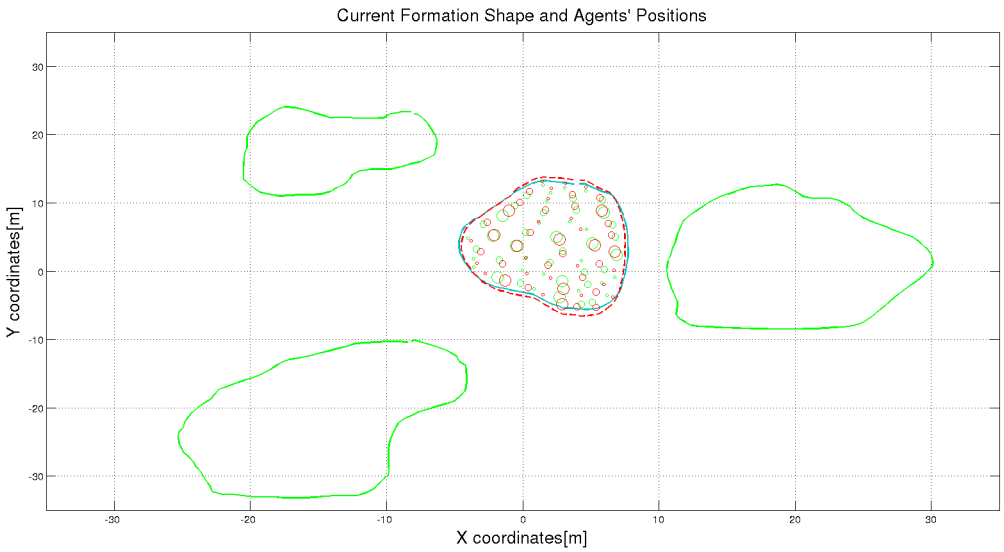


Figure 4.39: Latency in Bubble Packing Method-3

Figure 4.37, 4.38 and 4.39 compares the responses of these two algorithms to dynamically changing formation shapes. Bubble Packing and Artificial Force algorithms are executed with the same dynamical formation shapes and same initial positions of the agents. In these figures, red circles are representing the agents' positions with Bubble Packing method and green circles are representing the agents' positions with Artificial Forces method at the same time from the beginning of the simulations. Formation shape with blue line illustrates the current formation shape. It is obvious that the green agents are adapted to the current formation shape better than the red ones. The shape with dotted red line, represents the estimated coverage area of red agents in the environment and this shape differs from the current formation shape locally at some boundary regions. This shows that agents response to the dynamically changing formation shape faster with Artifical Forces method.

4.3.5 Evaluation

Three different formation control approaches are evaluated with different kinds of metrics including the total displacement, settling time, mesh quality and dynamically changing formations. Artificial forces method has the worst performance in settling time and total displacement metrics due to the absence of predetermined goal states

which is discussed in Section 4.3.3 in details. On the other hand the Randomized Fractals method has the worst mesh quality performance because of its randomized nature of assignment the goal states in the desired formation shape. It is not possible to adapt the agents for dynamically changing formations with this method as discussed in Section 4.3.4. Artificial forces method has better response to the dynamically changing formations than the Bubble Packing method. The methods which have the worst performance at the related metrics are illustrated in Table 4.3

Table 4.3: Formation Control Methods with Worst Performance

Method/Metric	**Total Displacement**	**Settling Time**	**Mesh Quality**	**Dynamical Formation**
Artificial Force	X	X		
Bubble Packing				X
Randomized Fractals			X	N/A

It is obvious that Bubble Packing Method has the best performance for all metrics except for dynamical formation shapes. In fact, Bubble Packing method combines the efficient approaches of the other two methods. In the shape partitioning phase of the algorithm, it uses interbubble forces which have a similar structure with the intermember forces applied by the Artificial Forces method in formation control. This force has the greatest effect on the homogeneity of the agents in the desired formation shape because it provides a global consensus for the agents in which each agent reaches an equilibrium state under the total force acting by their neighbors and the formation shape. On the other hand, Bubble Packing method implements an algorithm in which every agent is assigned to a goal state instantly at each execution period of the algorithm to minimize the overall displacement of the swarm. This approach reduce the settling time and the total displacements of the agents from the initial state to their final states since their final positions in the formation shape is predetermined. It is possible that the agents' goal states may be changed with the execution of the algorithm at different steps, but these assignments will converge to a unique subset while the agents are getting closer to the goal states.

Artificial forces method has better performance on dynamically changing formations because it directly adapts the individual control signals on the agents according to the

current formation shape. Bubble Packing method has a slower response to the dynamically changing formations due to its solution with two stages, shape partitioning and goal state assignment.

We have made some different formation shape trials by using these different methods. Following figures and tables, illustrates these trials and their performance metrics.

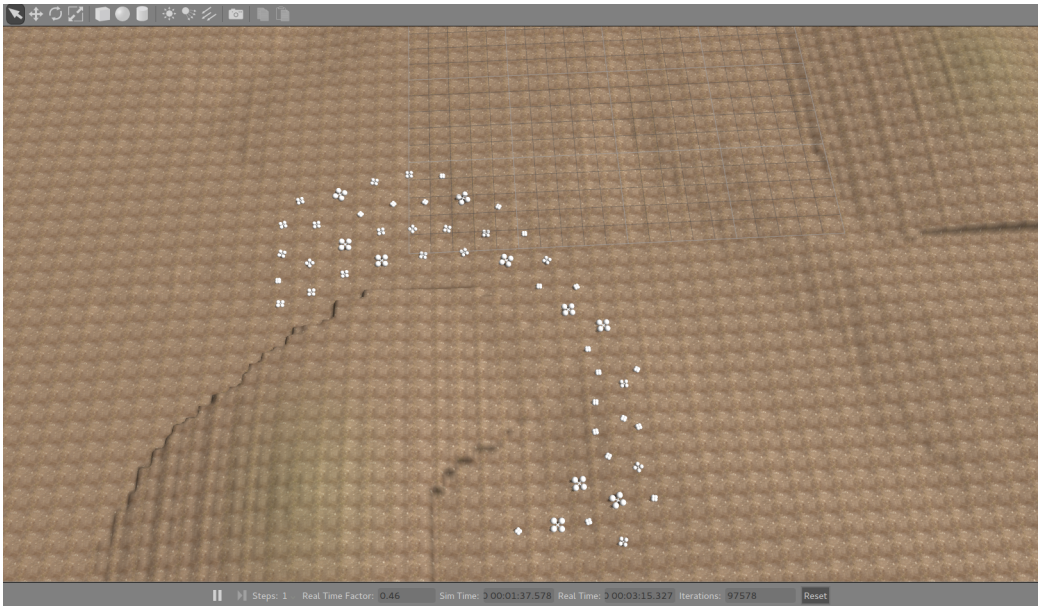


Figure 4.40: Formation Shape - 1 in Gazebo Environment

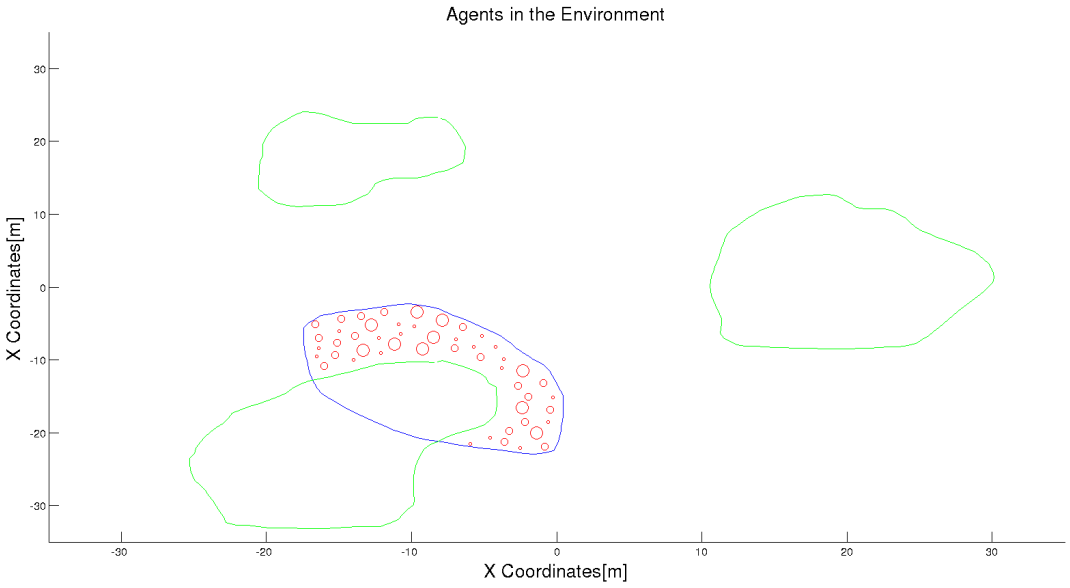


Figure 4.41: Formation Shape - 1 in MATLAB Environment

Table 4.4: Performance Metrics for Shape - 1

Method	**Total Displacement[m]**	**Total Settling Time[sec]**	ε_t	ε_g
Artificial Forces	1889	164	2.54	0.62
Bubble Packing	1750	151	2.44	0.67
Randomized Fractals	1762	153	3.14	0.92

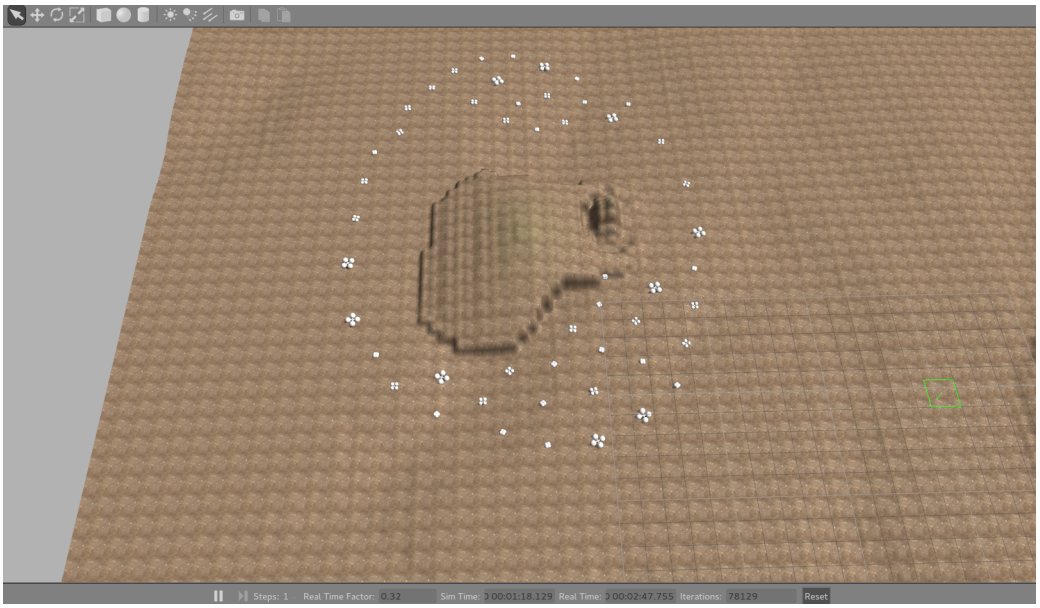


Figure 4.42: Formation Shape - 2 in Gazebo Environment

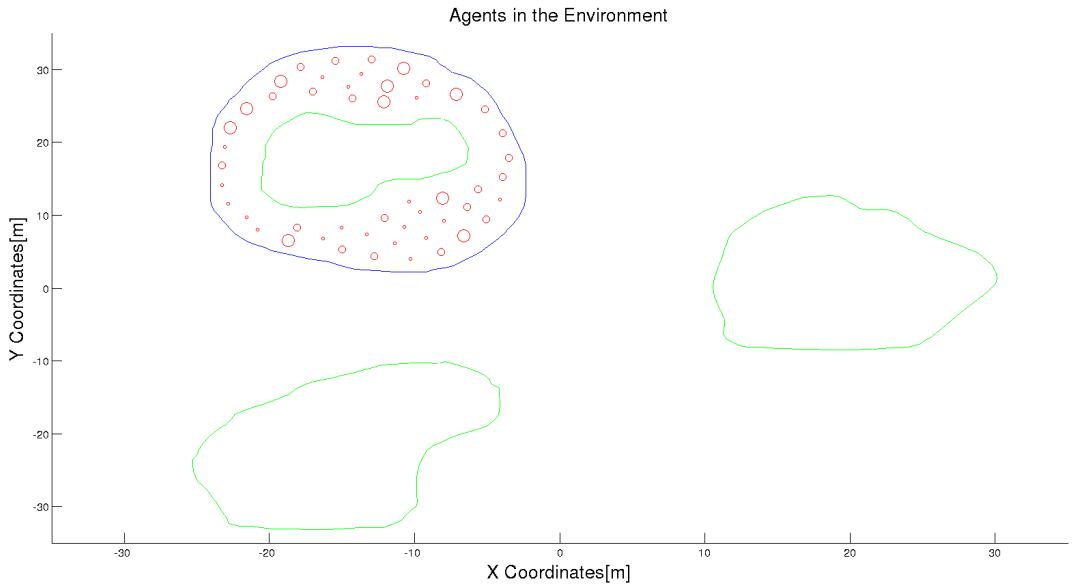


Figure 4.43: Formation Shape - 2 in MATLAB Environment

Table 4.5: Performance Metrics for Shape - 2

Method	**Total Displacement[m]**	**Total Settling Time[sec]**	ε_t	ε_g
Artificial Forces	1611	153	2.95	0.73
Bubble Packing	1550	141	2.84	0.77
Randomized Fractals	1522	140	3.24	1.12

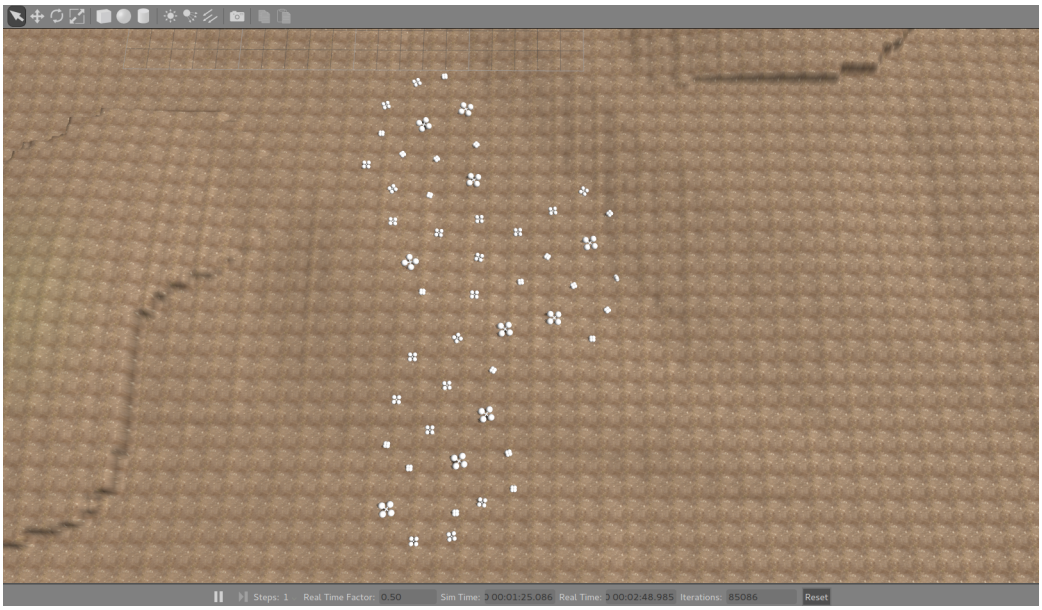


Figure 4.44: Formation Shape - 3 in Gazebo Environment

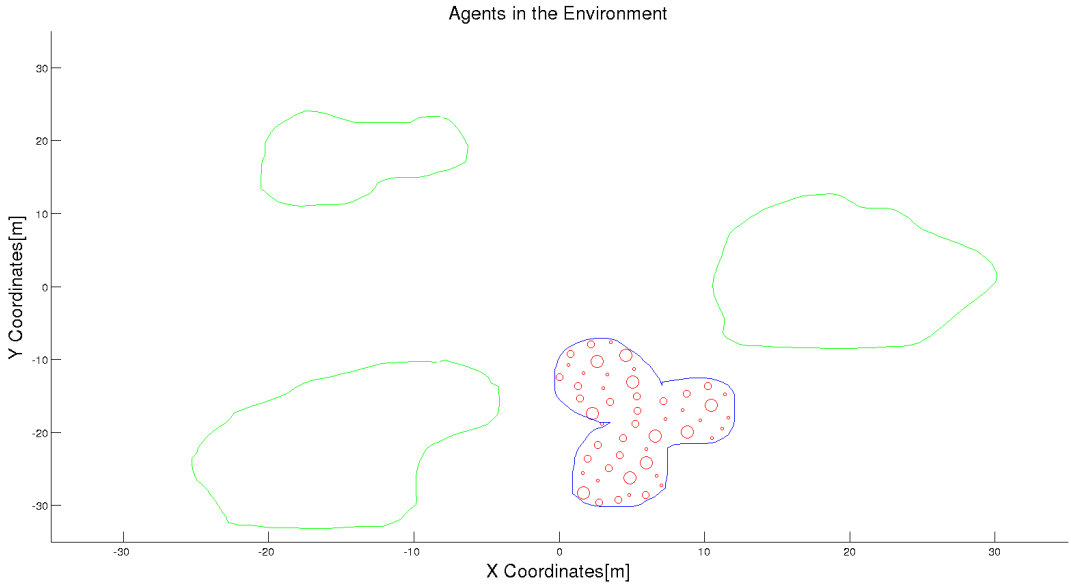


Figure 4.45: Formation Shape - 3 in MATLAB Environment

Table 4.6: Performance Metrics for Shape - 3

Method	Total Displacement[m]	Total Settling Time[sec]	ε_t	ε_g
Artificial Forces	1432	126	2.23	0.45
Bubble Packing	1330	115	2.14	0.52
Randomized Fractals	1350	112	3.15	0.89

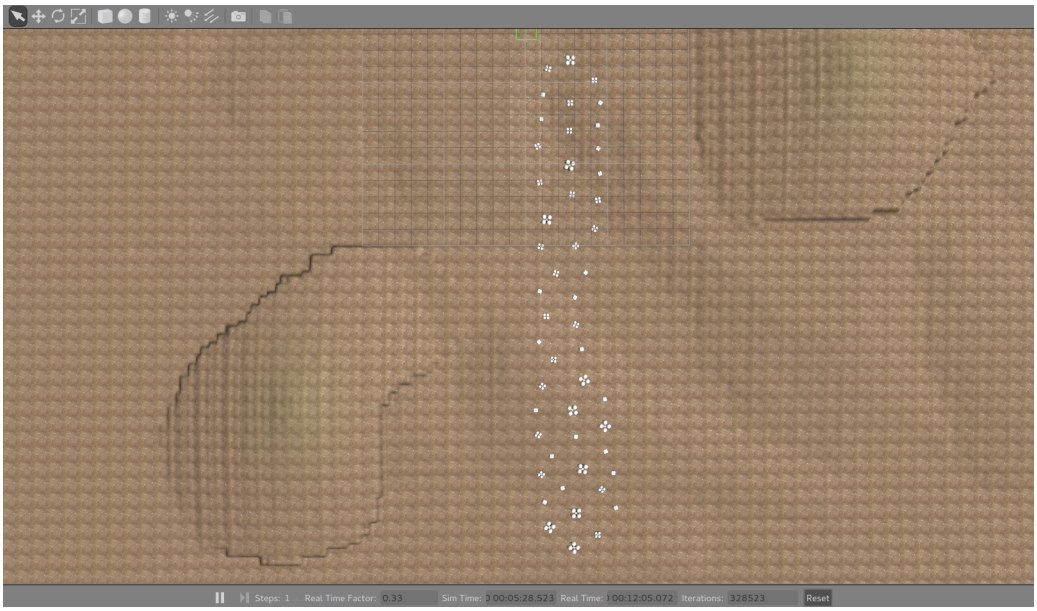


Figure 4.46: Formation Shape - 4 in Gazebo Environment

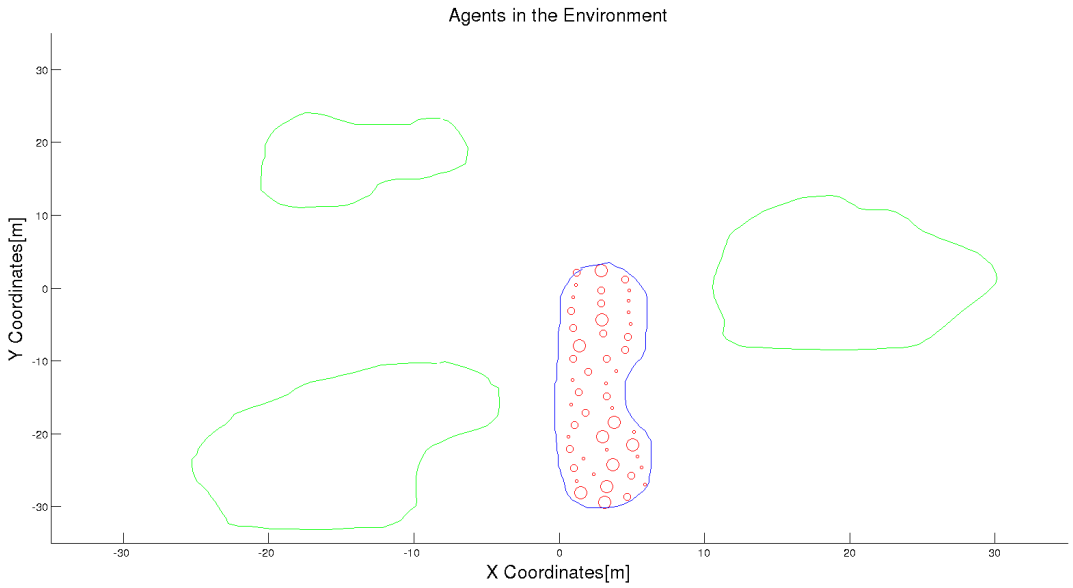


Figure 4.47: Formation Shape - 4 in MATLAB Environment

Table 4.7: Performance Metrics for Shape - 4

Method	**Total Displacement[m]**	**Total Settling Time[sec]**	ε_t	ε_g
Artificial Forces	1578	142	2.15	0.56
Bubble Packing	1345	112	2.34	0.66
Randomized Fractals	1321	121	3.23	1.05

A simulation with multiple formation shapes sequentially is illustrated in Figure 4.48.

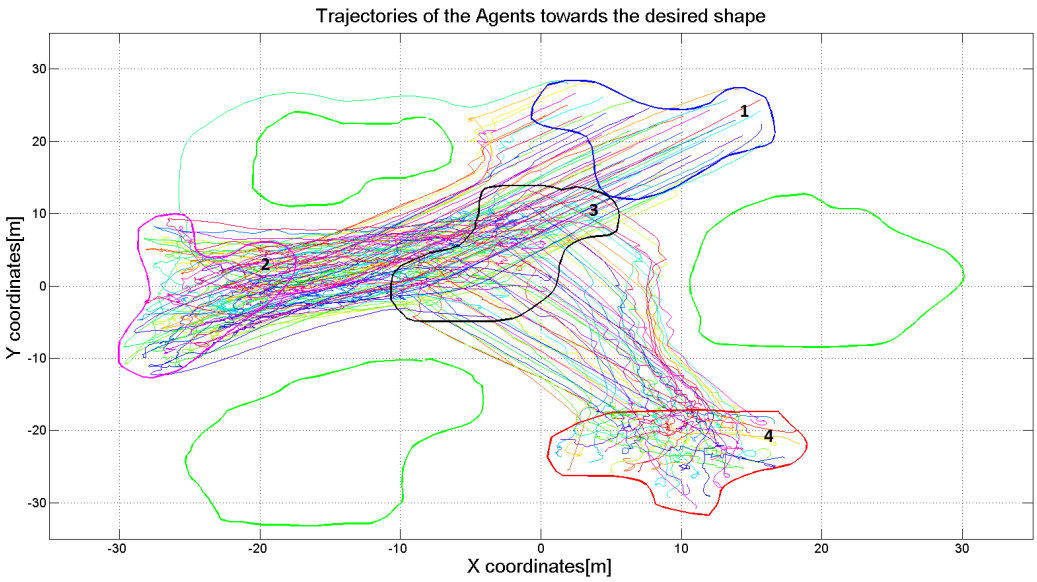


Figure 4.48: Multiple Formations

Table 4.8: Performance Metrics for Multiple Formation Shapes

Method	**Total Displacement[m]**	**Total Settling Time[sec]**	ε_t **Mean**	ε_g **Mean**
Artificial Forces	4216	435	2.21	0.68
Bubble Packing	3876	367	2.31	0.56
Randomized Fractals	3769	372	3.35	1.16

CHAPTER 5

HARDWARE IMPLEMENTATION & RESULTS

In this project, we have done a hardware demonstration with five mobile robots from different sizes, to achieve a proof of concept related with the topics we have focused on. In this implementation, agents are designed as mobile robots with three omni-wheels which allows them to navigate in the workspace to every direction without the need for changing their headings. We have designed both hardware and software systems of the mobile robots. Since the demonstration is held at indoor environment, we have implemented image processing algorithms to provide position information for each agent with the help of an E/O camera. Local positioning system is not implemented for this hardware demonstration because it may not be possible for an agent to find enough position beacons to localize itself in a swarm with five agents. Formation control algorithms are implemented with Bubble Packing method which has the best performance on different metrics described in 4.3.5

5.1 Hardware Demonstration Environment

The schematic of the hardware implementation environment is illustrated in Figure 5.1

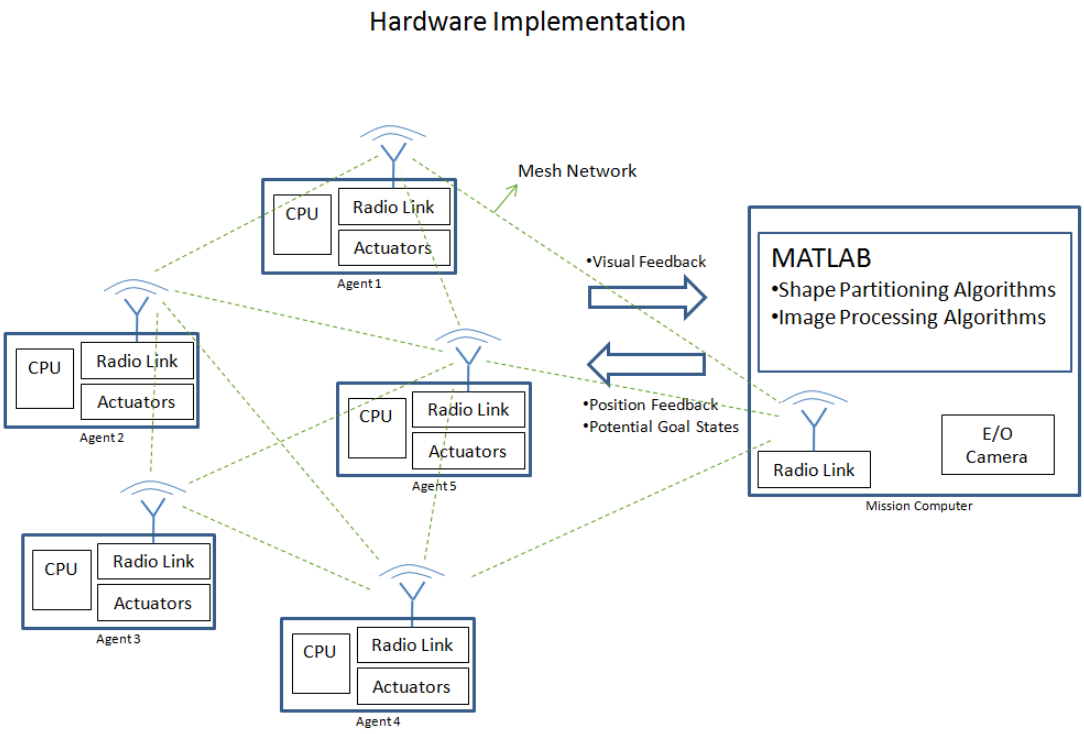


Figure 5.1: Implementation Environment

As illustrated in Figure 5.1, each agent and the mission computer has a radio link to create a mesh network in which every node can transfer messages between each other directly or with the help of their neighbors. Also they can broadcast messages to the rest of the nodes in the network. On the other hand each agent has their individual CPU. This processor unit is used to execute goal state decision process and to control the actuators of the agents. These processors also executes the control system which is described in Section 3.2.4 and they manage the messaging operations within the mesh network. This architecture supports the idea of decentralized formation control with Bubble Packing method in which each agent is responsible to take decisions and reach to a global consensus with the rest of the swarm on the potential goal states. These potential goal states are determined by the mission computer which takes the desired formation shape from the operator and executes shape partitioning algorithms. The data including the potential goal states is broadcasted to all of the agents in the environment with the help of mesh network.

Since the workspace is an indoor environment, it is impossible to use a GNSS system

to provide position measurements to the agents. To provide this external measurements, a visual feedback system is designed with the help of an E/O camera and image processing algorithms. We have implemented image processing algorithms depending on the color classification of different cover planes placed on the top of the mobile robots. These covers are illustrated in Figure 5.2

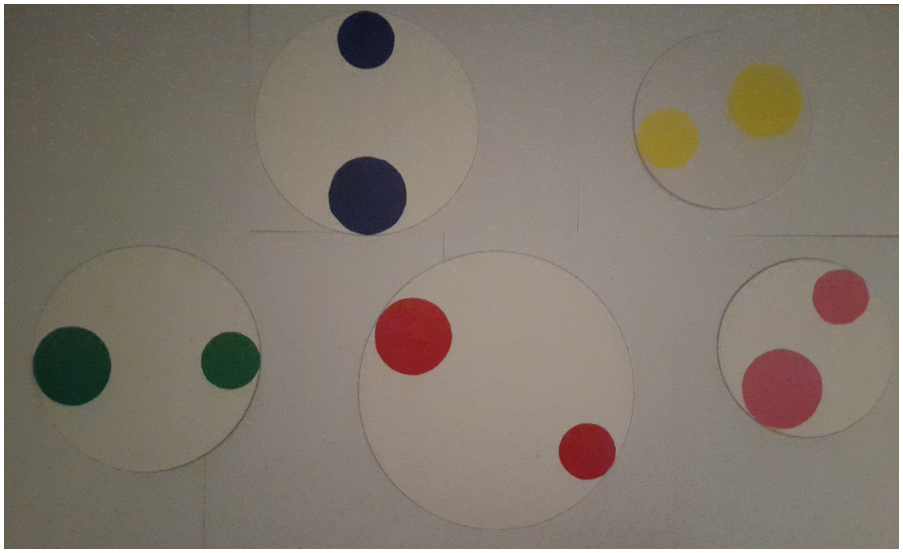


Figure 5.2: Covers for Different Types of Agents

Covers have different sizes to represent the coverage circles of the agents in 2D environment. These covers are used to create a swarm with heterogeneous agents partially. In our implementation, each cover has two different size of circles with the same color. Colors of these circles are used to classify the agent with the help of color classification algorithms. We have used different sized circles placed on the same cover, to calculate the heading angle of the agent with the help of circle detection algorithms. These algorithms can detect the positions and the radius of the circles in an image. In our project, heading angle of an agent is determined with the clockwise bearing angle of the vector from the center of the large circle to the center of the small circle. Figure 5.3 illustrates this calculation.

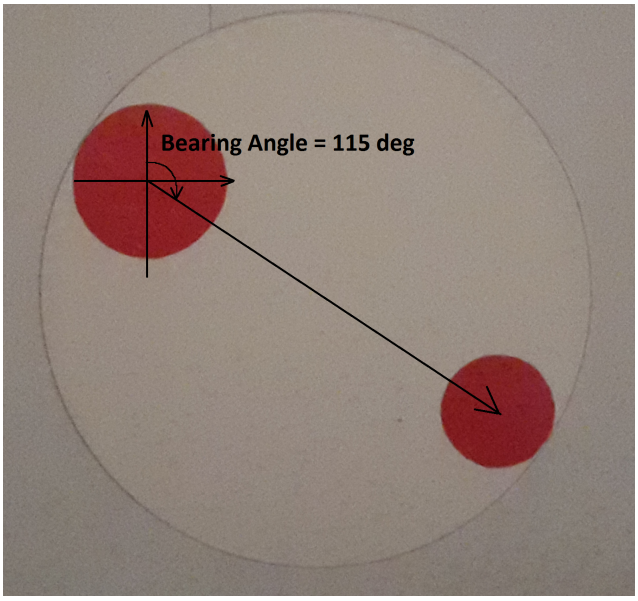


Figure 5.3: Orientation of an Agent in the Environment

The video of the environment is transmitted to the mission computer in nearly real time with the help of an E/O camera. Mission computer executes the image processing algorithms which filters the desired colors and detects the positions of the colored circles and broadcasts the position and heading angle datas of the agents. A sample output of the image processing algorithm is illustrated in Figure 5.4.

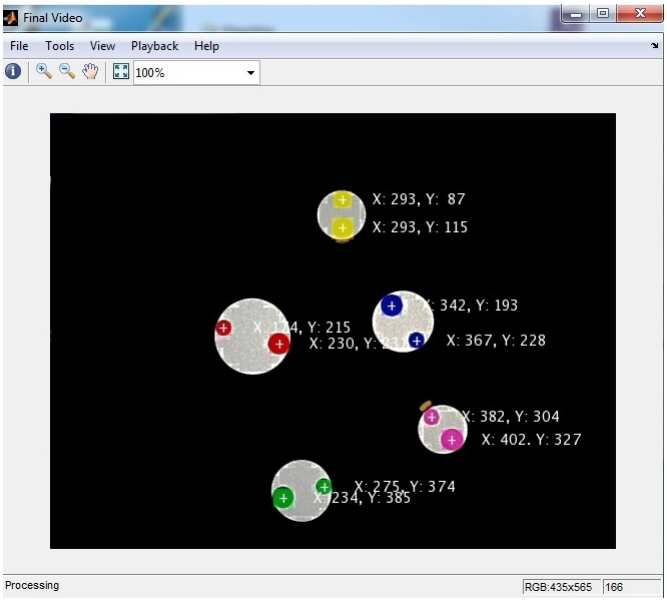


Figure 5.4: Sample Output of the Image Processing Algorithm

Each agent has its individual processor unit and radio link on its board. A block diagram of an individual agent's hardware is illustrated in Figure 5.5

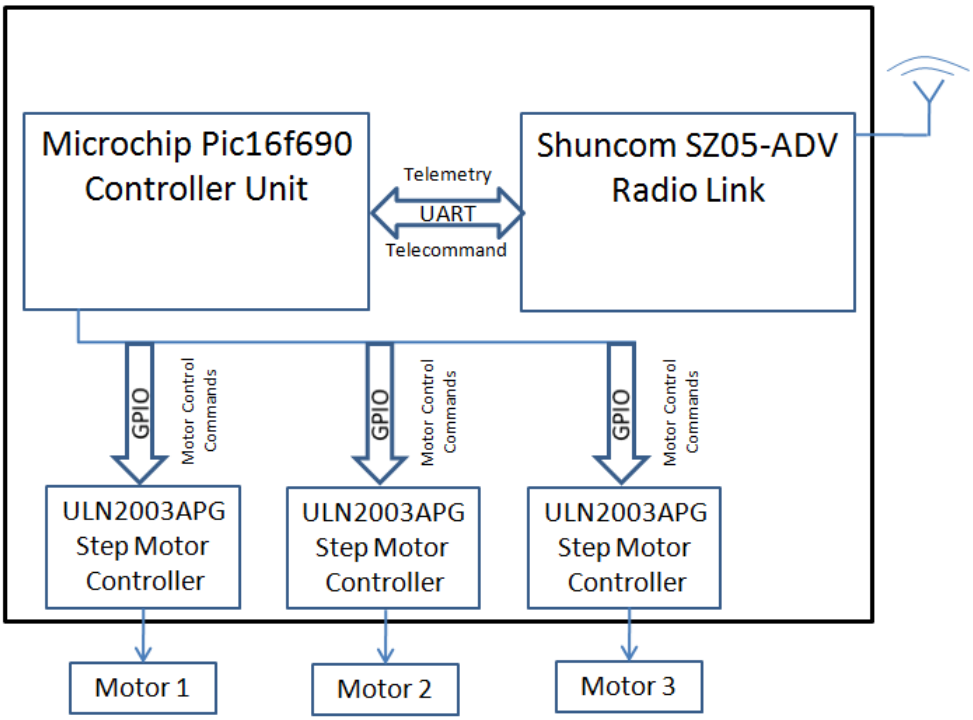


Figure 5.5: Hardware of an Agent

We use Microchip's Pic16f690 microcontroller as processor unit and write the related embedded software to drive the other peripherals on the board. This unit executes the decision process on formation control and the navigation control system defined in 3.2.4 and drives the 3 step motor control units, ULN2003APG, via GPIO peripherals. In our embedded software, instant rotational velocity setpoints for the stepper motors are determined with the command mixture algorithm illustrated in Figure 5.6.

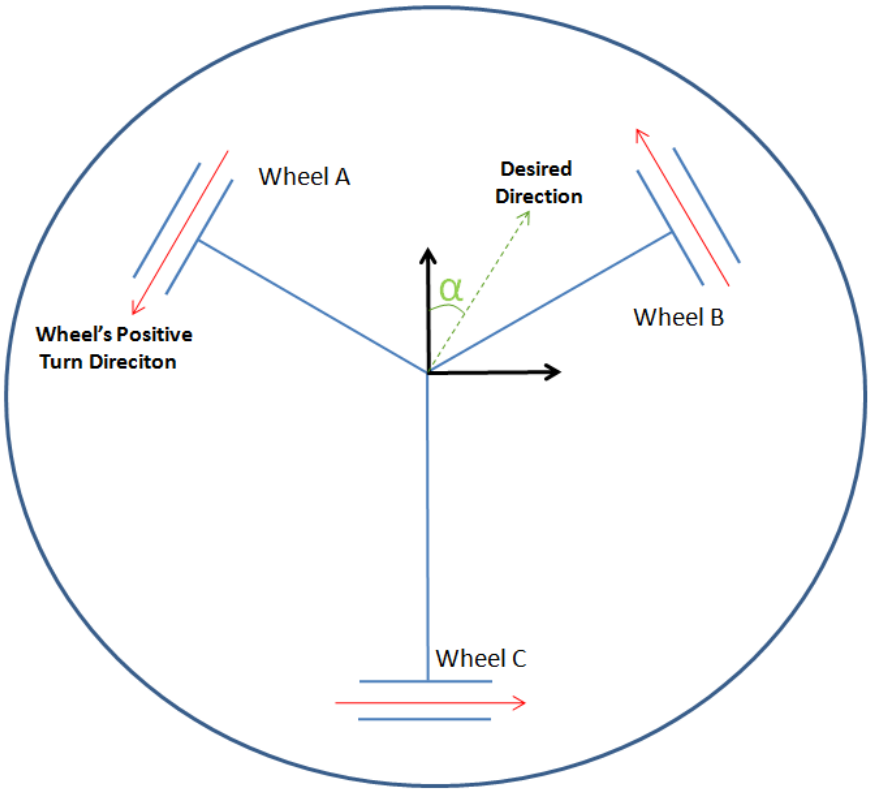


Figure 5.6: Command Mixture of Step Motors

The desired velocity vector for the agent is distributed to the stepper motors in accordance with the heading of the agent . Let $\|Vel\|$ be the amplitude of the desired velocity and α is the angle representing the desired direction of the movement in clockwise direction with respect to mobile robot's body frame. The velocities for the stepper motors is determined with following equations,

$$V_A = \|Vel\| \, cos(150 - \alpha)$$
$$V_B = \|Vel\| \, cos(30 - \alpha)$$
$$V_C = \|Vel\| \, cos(270 - \alpha)$$

where V_A, V_B, V_C represents the desired velocities of stepper motor A,B and C respectively.

Microcontroller also drives the radio link via UART peripheral and manages the com-

munication of the agent with the mesh network. All of the units on the board are supplied with a 5VDC regulator, 7805. The schematic of the circuit which controls the agent is illustrated in Figure 5.7 and 5.8.

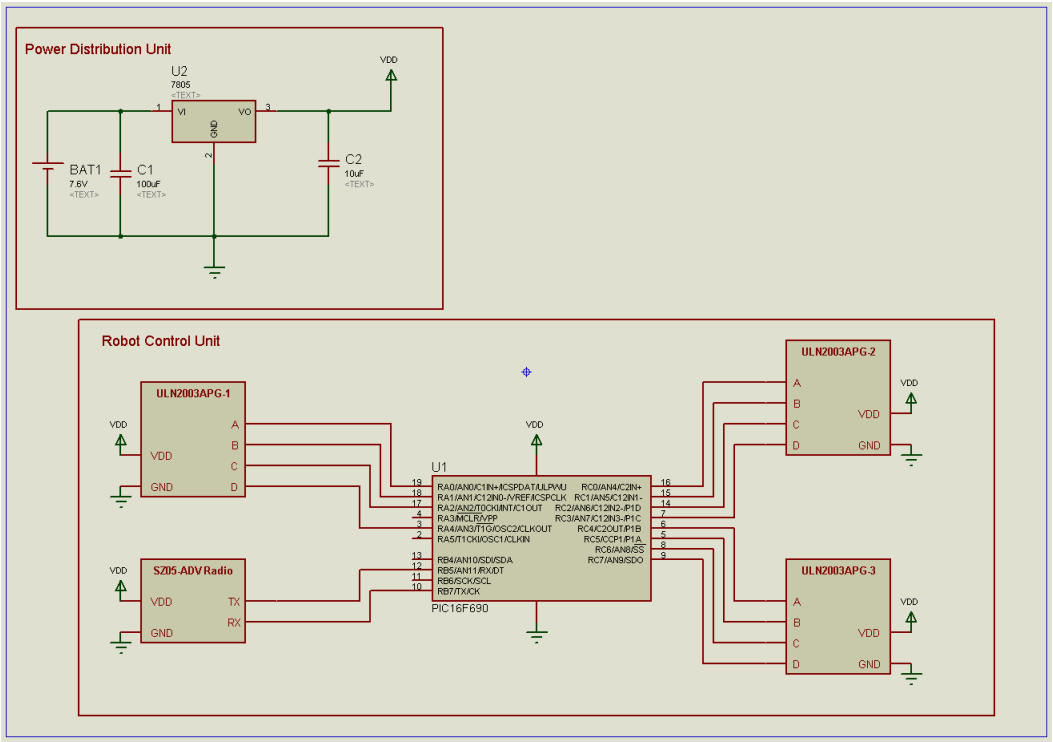


Figure 5.7: Schematic of the Circuit on the Board

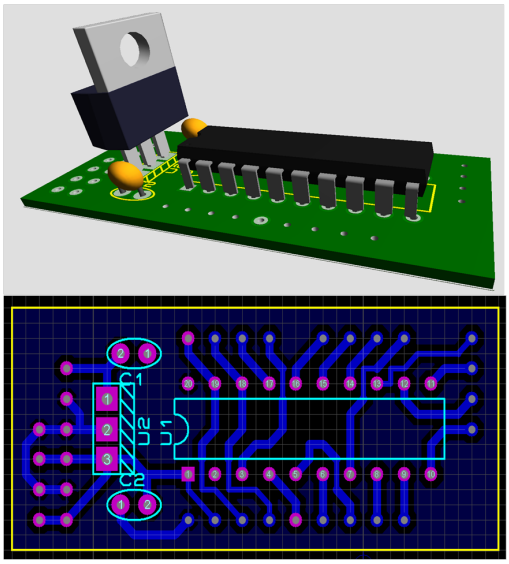


Figure 5.8: 3D Visualization of the Layout

Figure 5.9 and Figure 5.10 describes the hardware parts used on a mobile robot.

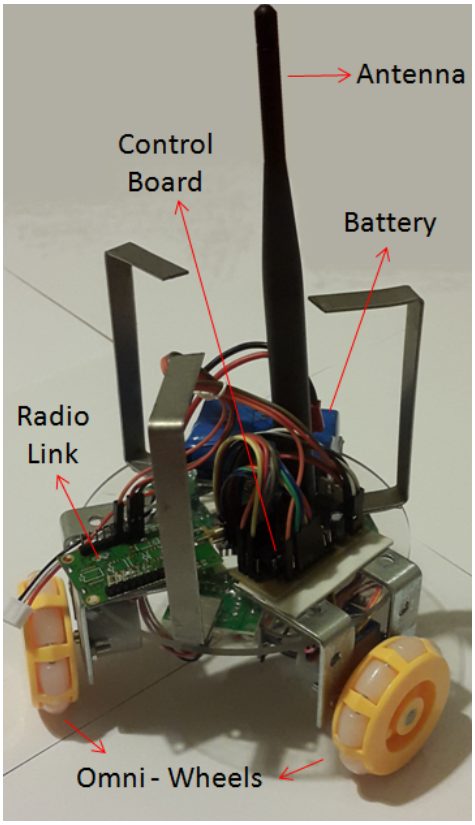


Figure 5.9: Hardware of an Agent - Top View

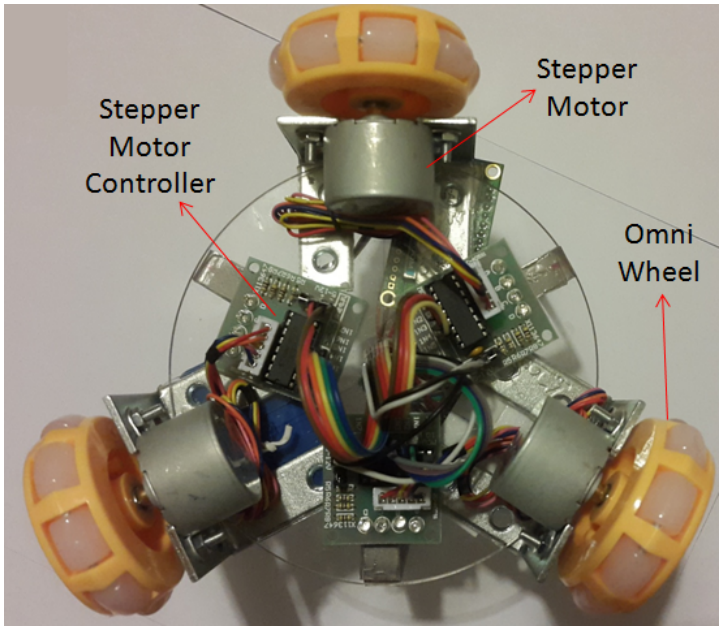


Figure 5.10: Hardware of an Agent - Bottom View

5.2 Performance Analysis

The system is tested with several formation shapes and results are analyzed with total displacements of the agents and the settling time metrics. Sample formation shapes are covered and the algorithm proposed for Bubble Packing method is tested in real time successfully. Traces of the agents are plotted with the continuous blue lines from their initial positions to the goal states in the following figures. The desired formation shape is plotted with black circles. The environment is a square area which has 2x2 meter size and there are three different types of agents which have different coverage circles represented in Table 5.1

Table 5.1: Three Different Agent Configuration

Agent Color	Agent Type	Coverage Circle Radius[cm]
Red	1	16
Blue	2	11
Green	2	11
Yellow	3	8
Pink	3	8

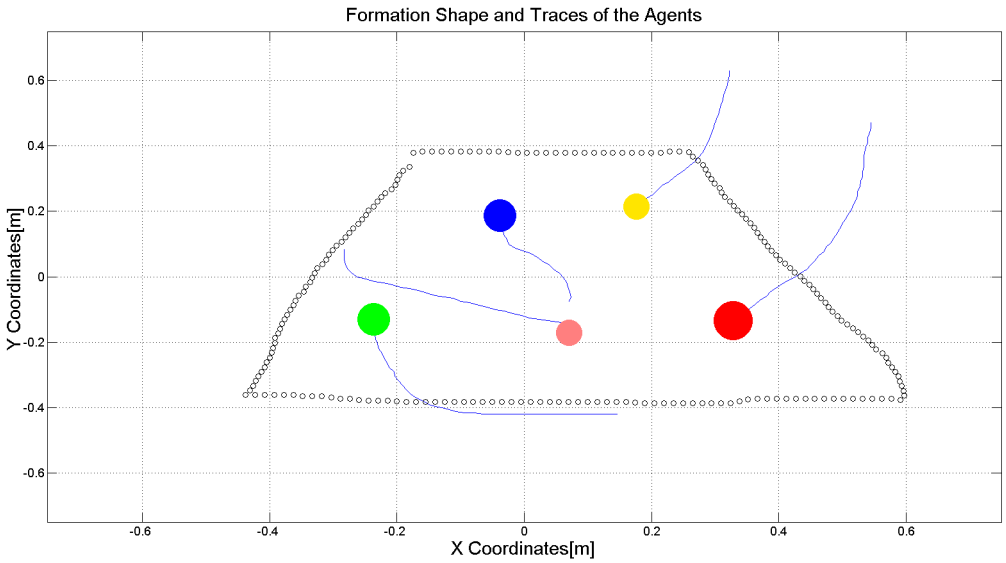


Figure 5.11: Formation Shape 1- Matlab Environment

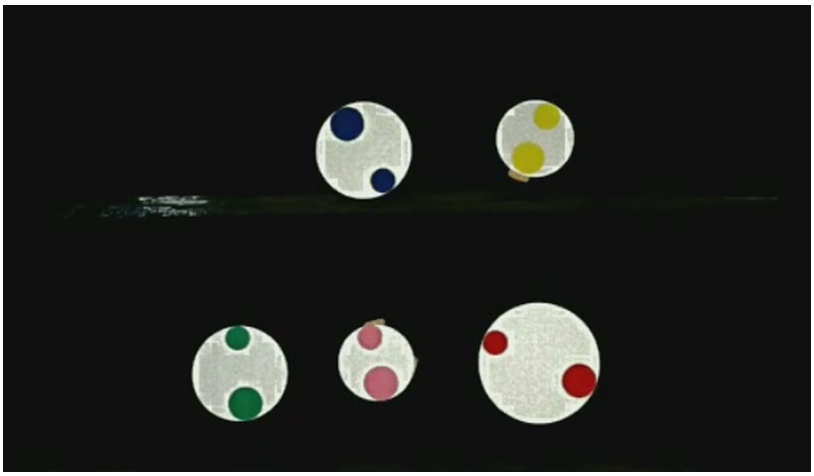


Figure 5.12: Formation Shape 1- Test Environment

Table 5.2: Performance Metrics for Shape - 1

Total Displacements[m]	**Settling Time[sec]**
1.75	23

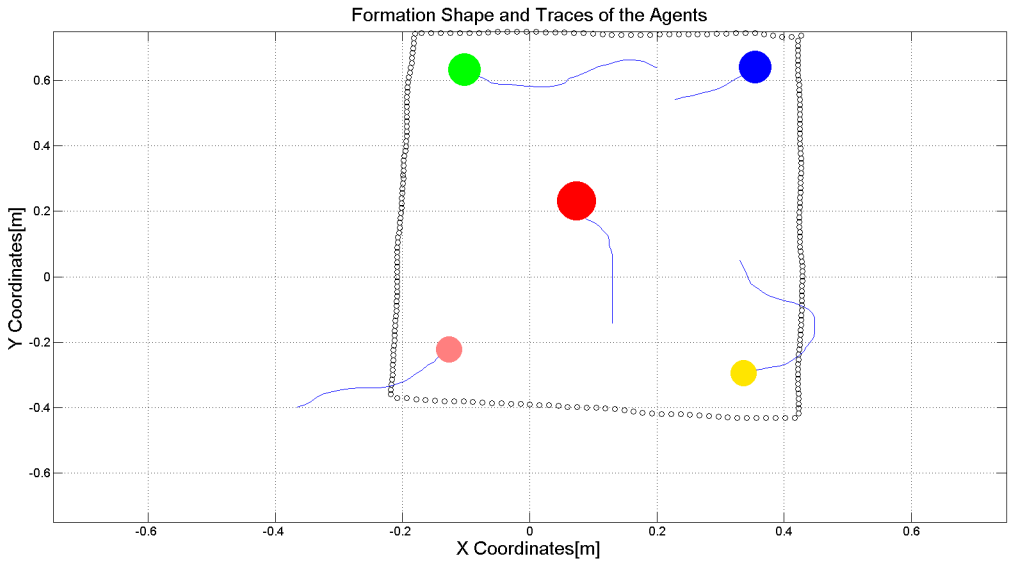


Figure 5.13: Formation Shape 2- Matlab Environment

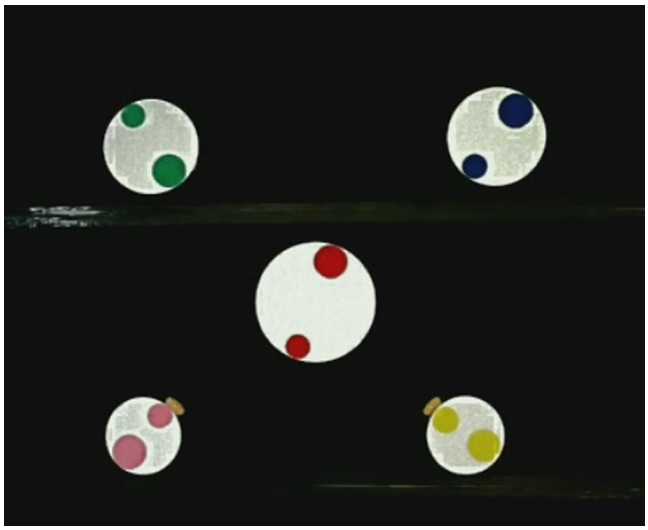


Figure 5.14: Formation Shape 2- Test Environment

Table 5.3: Performance Metrics for Shape - 2

Total Displacements[m]	**Settling Time[sec]**
1.63	19

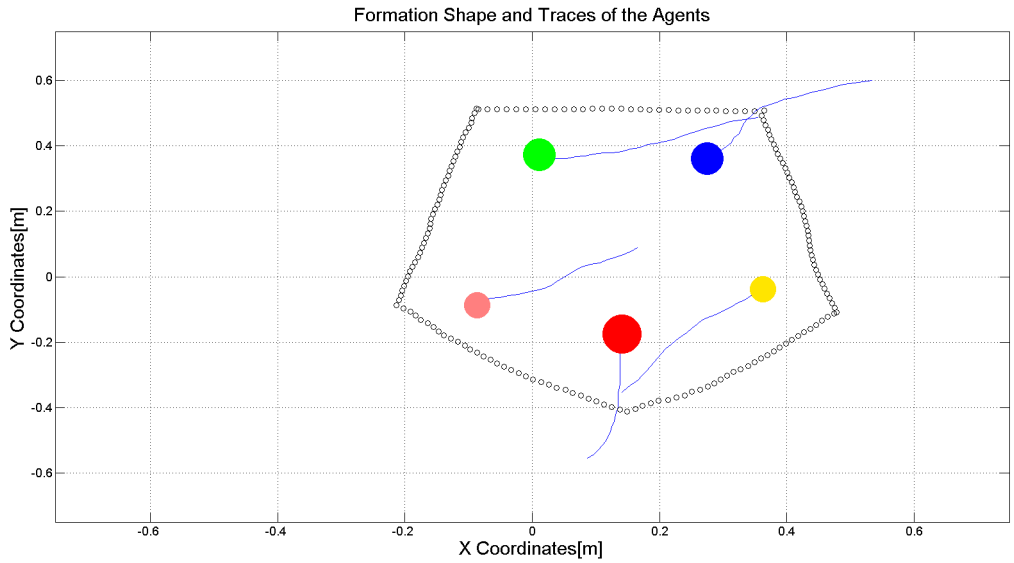


Figure 5.15: Formation Shape 3- Matlab Environment

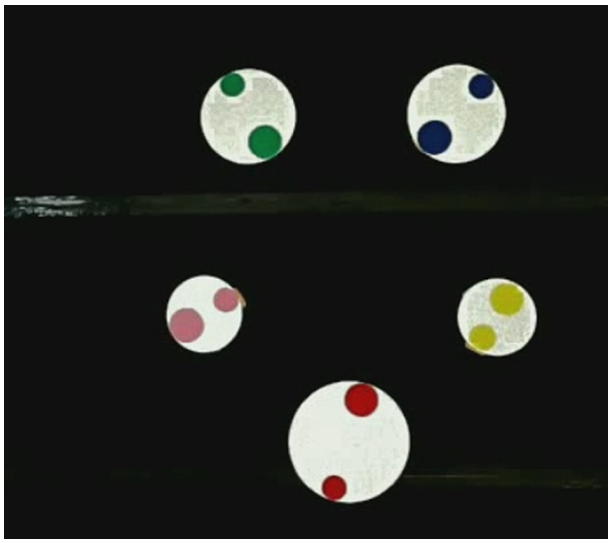


Figure 5.16: Formation Shape 3- Test Environment

Table 5.4: Performance Metrics for Shape - 3

Total Displacements[m]	**Settling Time[sec]**
1.95	26

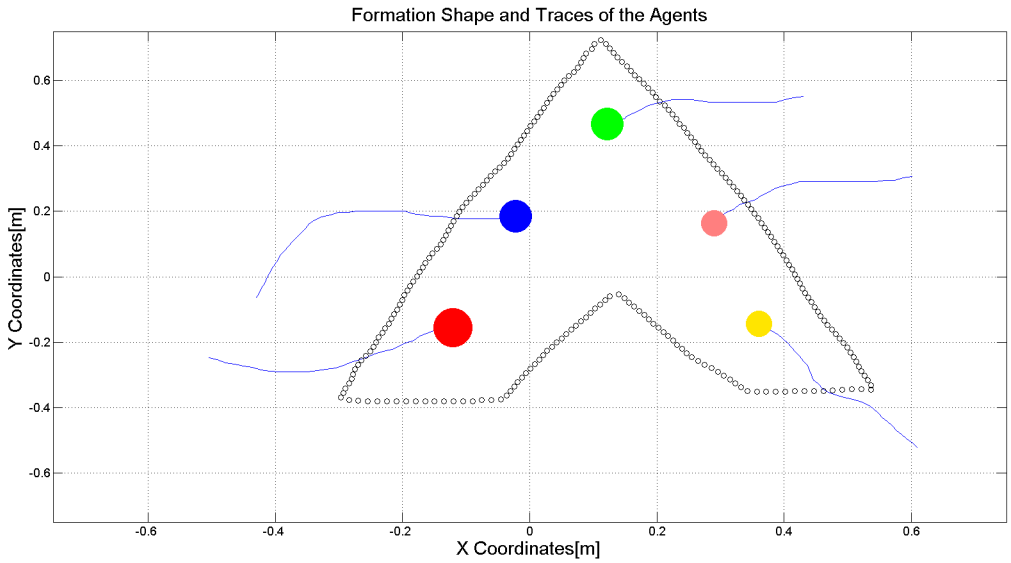


Figure 5.17: Formation Shape 4- Matlab Environment

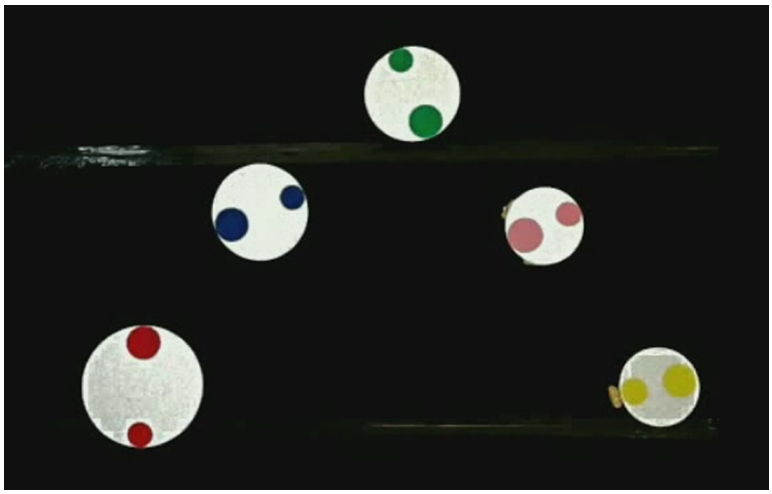


Figure 5.18: Formation Shape 4- Test Environment

Table 5.5: Performance Metrics for Shape - 4

Total Displacements[m]	**Settling Time[sec]**
2.16	29

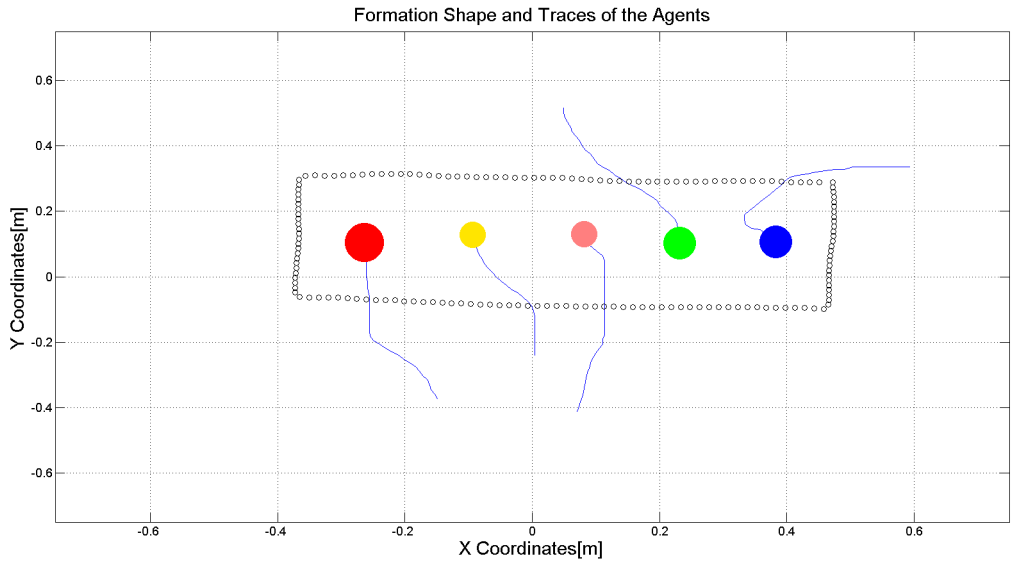


Figure 5.19: Formation Shape 5- Matlab Environment

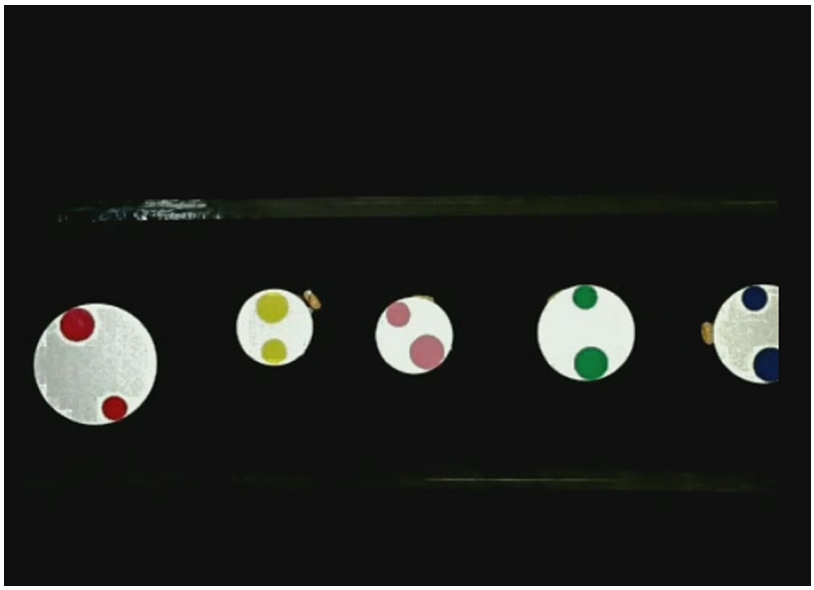


Figure 5.20: Formation Shape 5- Test Environment

Table 5.6: Performance Metrics for Shape - 5

Total Displacements[m]	**Settling Time[sec]**
1.80	25

We have defined the settling time as delta time "$t_{final} - t_{start}$" ,where t_{start} is the initial time and t_{final} is the time that the all of the agents are inside of the desired formation shape and the norm of the velocity vector for each agent $\|v_i\|$ is

$$\|v_i(t)\| < 0.01 \qquad [m/sec] \quad \forall t > t_{final} \tag{5.1}$$

Desired formations are simple geometrical shapes since it will not be possible to cover more complicated shapes with five agents. In Bubble Packing algorithm, goal states $g_i \in G$ are determined with the help of shape partitioning algorithm. This algorithm partitions the desired formation shape into goal states which are classified according to different agent types as we have discussed previously. In this demonstration we have implemented 3 different types of agents, so we have 3 different type goal states after each shape partitioning process. Red agent is the only type '1' agent in our system. So it always tries to reach the goal state which is dedicated to itself in different formation shapes given above. On the other hand, shape partitioning algorithm provides 2 goal states for type '2' agents which are blue and green ones. Decision

process on the goal states, assign these to agents to these 2 goal states to minimize the total displacement of the agents. Since we have 2 goal states, the assignment process has only two options in our case. It is possible to see that if the assignment of type '2' agents is switched (i.e. blue and green agents reach the other goal points rather than the current ones), total displacement will increase. This shows us the decision process works successfully in the assignment of the agents to minimize the total displacement. Similarly, we have 2 type '3' agents which are yellow and pink ones. It is possible to ibserve the same results for type '3' agents as we have discussed on type '2' agents.

CHAPTER 6

CONCLUSION AND FUTURE WORKS

6.1 Conclusion

In this thesis work, we have proposed solutions to the dynamic formation control of heterogeneous mobile robots. These solutions are constructed with the help of two different subsystems, local positioning systems and formation control systems.

A local positioning system is designed to provide agents accurate position and velocity data. It is assumed that only a percentage of the swarm has position measurement sensors (e.g. a GNSS solution) due to the low sensor capabilities which is discussed in Section 3.1. So there is a necessity to implement a solution for estimating the positions and the velocities of the rest of the agents which do not have external position measurement sensors on their board. A solution is proposed including the fusion of inertial measurements of the agents and the position data calculated with the help of the position beacons (i.e. the agents which have position measurement sensors on their boards). An algorithm that provides INS with the help of IMU sensors (i.e. Accelerometers, Gyroscopes, Magnetometers) is designed for each agent which propagates the translational accelerations of the agents in the Earth frame by implementing an AHRS system. Since this INS solution is vulnerable in position and velocity estimations due to the drift problems and measurement noise of the sensors, an external measurement of position is provided to each agent to correct their estimations based on local trilateration process. Local trilaterion process includes the distribution of the position data from the position beacons to the agents which do not have position sensors. Since it is assumed that the agents have limited communication capabilities, a

solution to distribute the position data with minimum error to the agents farthest from the position beacons is provided with the help of route tables and assignment of rank values to the agents in the mesh network. Trilateration process is handled with the local neighbors of each agent to calculate the estimated position data. This position data is used in the Kalman observers as the external measurement data. Since it is not possible to execute the route table determination and the local trilateration process with the main process period of the agents (which controls the actuators, propagates INS measurements etc.) due to the computational complexity of the algorithm, a maximum period of time to execute localization process is determined with the help of an upper error bound allowable to achieve formation control with success. The maximum error norm for each agent is calculated below 3 meters with Monte Carlo simulations with the localization period of 3 seconds. This error value is chosen as the maximum tolerable value for the formation control system. So the period for the localization timer is chosen as 3 seconds.

In the second subsystem of the solution, we have introduced three types of methods based on potential field approach and shape partitioning approaches. Potential field approach implements virtual forces to the agents created by the desired formation shape, obstacles in the environment and the other agents. The X-Swarm theory is introduced to prove that the movement of an agent which is outside of the desired formation shape will be towards to the center of the swarm. The potential field based algorithm is designed to satisfy the conditions for the X-Swarm to direct the agents to the center of the swarm which is also moving towards to the center of the formation shape. It is observed that the agents are capable of homogeneously covering of the desired formation shape with the help of these artificial forces.

The other two methods based on shape partitioning approaches, are Bubble Packing method and randomized fractals method . These methods have different solutions to partition the desired formation shape into potential goal states. The algorithms which determines the assignment of the agents to these goal states in a decentralized manner is identical for these two methods.

Bubble Packing algorithm introduces a method to partition the desired formation shape based on interbubble forces, similar to the Van der Waals forces between the

molecular bonds, to distribute the bubbles homogeneously. It is widely used in mesh generation problems. The idea is to generate a mesh for a surface with identical bubbles to mimic a regular Voronoi diagram including the vertices representing the center of the bubbles. In our problem, the agents are represented by the bubbles with equal radius' of their coverage circles described in Section 3.2.1.1, and the algorithm is executed to generate the desired homogenous mesh to partition the shape into potential goal states. Each goal state has the information of agent type (i.e. coverage circle radius) to be placed at the final configuration. The Bubble Packing method have a similar approach with the potential fields since they both implement intermember&interbubble forces to homogeneously distribute the agents&bubbles in the desired formation&surface

Randomized fractals method is based on randomly distributing the coverage circles which represent the different agent types in the desired formation shape. This algorithm is faster than the Bubble Packing method since it randomly determines potential goal states. Because of this randomized approach, it is not applicable to implement a dynamic formation control solution with this method.

The assignment process of the agents to the potential goal states is identical for these two different shape partitioning methods. First, the obstacles in the environment is augmented with Minkowski sums for different types of agents. The total coverage of the augmented obstacles represents the forbidden space for each agent. The free configuration space is calculated by extracting these forbidden spaces from the configuration space itself. Then each agent calculates the visibility graphs in the environment with augmented obstacles by including its current position and the potential goal states determined by the shape partitioning algorithms. Shortest paths in these visibility graphs to the each potential goal states are calculated with the help of Dijkstra's Algorithm. Hungarian algorithm is implemented to reach a global consensus on the assignment of the agents to the goal states based on minimizing the total displacement of the swarm by taken into consideration the distances of shortest paths to each goal states reported by each agent.

Monte Carlo simulations with 1000 iterations is held with the same initial conditions and formation shapes for the three different proposed solution. To evaluate the perfor-

mance of these methods, mesh quality, total displacement, settling time and dynamic formation performance metrics are calculated. As expected the worst mesh quality performance belongs to the randomized fractals method since it has an approach to randomly distribute the coverage circles in the desired formation shape. The potential field based method and the Bubble Packing method have similar mesh qualities because of their nature of implementing intermember&interbubble forces which contributes the homogeneously distribution of the agents&bubbles in the desired formation shape. On the other hand, potential field based method has the worst settling time and total displacement performance due to the lack of predetermined goal states for the agents in the desired formation shapes. The best performance is achieved with the Bubble Packing method in terms of these three metrics. Hence some different formation shape trials including special cases like a formation shape which includes obstacles are held with Bubble Packing method.

At the end of the work, a hardware implementation of Bubble Packing method is held with 5 mobile robots from different sizes. Since this implementation is done at the indoor environment and the number of agents is not sufficient, it is not possible to implement the local positioning system at this demonstration. The position and orientation data for the agents are calculated with image processing algorithms based on visual feedback provided by an E/O camera. A mesh network established with Zigbee modules and these modules are used to provide a communication backbone between the agents and the mission computer. The agents are designed as 3 axis omni-wheel mobile robots which have the capability to change position in the environment without the need for adapting their orientation. Each agent has its own processor module to execute the goal state decision process, control algorithms and step motor control routines. This feature demonstrates the decentralized manner of the solution in which there is no central server deciding the final states of each agent in the solution. Several formation shape trials are done and the performance is evaluated with total displacement and settling time metrics. It is observed that the agents are capable to achieve different formation shapes as desired. This work is important to show the proposed solution about the formation control problem can be implemented in real time environment. The results represent a proof of concept (POC) of the thesis work rather than implementing the all details of the proposed solution.

6.2 Future Works

It is important to realize that there is a need to make hardware implementation with more agents within the environment. This kind of implementation will give a more insight about the performance and the drawbacks about the system according to the tests held in simulation environment. So, the implementation with more number of agents in real time is one of the next steps for this project.

Even if the Bubble Packing method has the best performance in the sense of both mesh quality, total displacement and settling time issues, it is not a fully decentralized method since the shape partitioning process is executed on a central server node. Shape partitioning algorithm is not deterministic and it doesn't converge to the same potential goal states even if the initial states are identical. If it is possible to implement a method to partition the desired formation shape with a similar mesh quality performance in a more deterministic manner, it will be possible to distribute the partitioning process to individual agents.

Since the main focus is not on the obstacle avoidance issue of the agents, all three methods are augmented with only potential fields created by the obstacles in the environment just to implement a basic obstacle avoidance. It is needed to implement a more complex obstacle avoidance algorithm (e.g. tangent bug algorithm) to avoid the cases in which some of the agents get into unwanted equilibriums under the effect of the desired setpoints and the obstacle forces.

In this thesis work, one of the main metrics to evaluate the performance of the proposed solutions is the total displacement of the swarm while achieving the desired formation shape. Moreover, in the shape partitioning methods the goal state assignment algorithms are designed to minimize the total displacement of the swarm. But, there may be cases in which some of the agents are more critical to reach the goal state or some of the agents are running out of battery&energy. In such conditions, it will be appropriate to improve the goal state assignment algorithm to prioritize the agents to handle these kind of special cases.

The main aim was to create a swarm with heterogeneous agents which can achieve a desired complex formation. One of the idea about this concept is to achieve some

complex tasks collectively with the agents which have different individual capabilities. In our thesis work, we have focused on achieving complex formation shapes with heterogeneous mobile robots rather than achieving complex tasks after getting the desired formation shapes. The next step for this project is to add the capability of doing these kind of tasks with the agents from different capabilities.

REFERENCES

[1] Jan Barca and Ahmet Sekercioglu. Swarm robotics reviewed. *Robotica*, 31:345–359, 2013.

[2] Simon Garnier, Jacques Gautrais, and Guy Theraulaz. The biological principles of swarm intelligence. *Swarm Intelligence*, 1:3–31, 2007.

[3] Gerardo Beni. From swarm intelligence to swarm robotics. In *International Conference on Swarm Robotics*, 2004.

[4] Yogeswaran Mohan and S.G. Ponnambalam. An extensive review of research in swarm robotics. In *World Congress on Nature & Biologically Inspired Computing, NaBIC*, 2009.

[5] Aleksis Liekna and Janis Grundspenkis. Towards practical application of swarm robotics: Overviewed of swarm tasks. In *13th International Scientific Conference - Engineering for Rural Development*, 2014.

[6] Marco Dorigo, Vito Trianni, Erol Sahin, Roderich Grof, Thomas Labella, Gianluca Baldassarre, Stefano Nolfi, Jean Deneubourg, Francesco Mondada, and Dario Floreano. Evolving self-organizing behaviors for a swarm-bot. *Autonomous Robots*, 17:223–245, 2004.

[7] S.P. Hou, C.C. Cheah, and J.J.E. Slotine. Dynamic region following formation control for a swarm of robots. In *ICRA*, 2009.

[8] Ganesh Venayagamoorthy, Lisa Grant, and Sheetal Doctor. Collective robotic search using hybrid techniques: Fuzzy logic and swarm intelligence inspired by nature. *Engineering Applications of Artificial Intelligence*, 22:431–441, 2009.

[9] Luiz Chaimowicz, Mario Campos, and Vijay Kumar. Dynamic role assignment for cooperative robots. In *International Conference on Robotics and Automation*, 2002.

[10] Seyed Zekavat, Hui Tong, and Jindong Tan. A novel wireless local positioning system for airport (indoor) security. In *SPIE*, 2004.

[11] Kiattisin Kanjanawanishkul. Formation control of mobile robots: Survey. n.d.

[12] Dorigo et al. Swarmanoid: A novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics & Automation Magazine*, 20:60–71, 2013.

[13] Maurice Martin and Steve Kilberg. Techsat 21 and revolutionizing space missions using microsatellites. 2001.

[14] YangQuan Chen and Zhongmin Wang. Formation control: A review and a new consideration. In *International Conference on Intelligent Robots and Systems*, 2005.

[15] Michael Rubenstein, Adrian Cabrera, Justin Werfel, Golnaz Habibi, James Mclurkin, and Radhika Nagpal. Collective transport of complex objects by simple robots: Theory and experiments. In *nternational Conference on Autonomous Agents and Multi-Agent Systems*, 2013.

[16] S Kornienki, O. Kornienko, and Levi. P. Minimalistic approach towards communication and perception in microrobotic swarms. In *IEEE International Conference on Intelligent Robots and Systems*, 2005.

[17] Farshad Arvin, John Murray, Licheng Shi, Chun Zhang, and Shigang Yue. Development of an autonomous micro robot for swarm robotics. In *IEEE International Conference on Mechatronics and Automation*, 2014.

[18] Touraj Soleymani, Vito Trianni, Michael Bonani, Francesco Mondada, and Marco Dorigo. Bio-inspired construction with mobile robots and compliant pockets. *Robotics and Autonomus Systems*, 74:340–350, 2015.

[19] Roderich Grof, Michael Bonani, Mondada Francesco, and Marco Dorigo. Autonomous self-assembly in a swarm-bot. 22:1115–1130, 2006.

[20] Reza Haghighi and Chien Chern Cheah. Asynchronous dynamic multi-group formation for swarm robots. In *50th IEEE Conference on Decision and Control and European Control Conference*, 2011.

[21] Magnus Egerstedt and Xiaoming Hu. Formation constrained multi-agent control. *IEEE Transactions On Robotics And Automation*, 17:947–951, 2001.

[22] Samitha Ekanayake and Pubudu Pathirana. Formations of robotic swarm: An artificial force based approach. *International Journal of Advanced Robotic Systems*, 7:173–190, 2010.

[23] Aveek Das, Rafael Fierro, Vijay Kumar, James Ostrowski, John Spletzer, and Camilla Taylor. A vision-based formation control framework. *IEEE Transactions on Robotics and Automation*, 18:813–825, 2002.

[24] Anthony Lewis and Kar-Han Tan. High precision formation control of mobile robots using virtual structures. *Autonomous Robots*, 4:387–403, 1997.

[25] Jimming Cheng, Winston Cheng, and Radhika Nagpal. Robust and self-repairing formation control for swarms of mobile agents. In *AAAI 20th National Conference on Artificial Intelligence*, 2005.

[26] Calin Belta and Vijay Kumar. Abstraction and control for groups of robots. *IEEE Transactions on Robotics and Automation*, 20:865–875, 2004.

[27] John Shier and Paul Bourke. An algorithm for random fractal filling of space. *Computer Graphics Forum*, 32:89–97, 2013.

[28] Kenji Shimada and David Gossard. Bubble mesh: Automated triangular meshing of non-manifold geometry by sphere packing. In *ACM Symposium on Solid Modeling and Applications*, 1995.

[29] Ioannis Rekleitis, Philippe Babin, Sourav Das, and Olivier Falardeau. Experiments in quadrotor formation flying using on-board relative localization. n.d.

[30] Martin Vossiek, Peter Gulden, Jan Wieghardt, and Clemens Hoffman. Wireless local positioning - concepts, solutions, applications. In *RAWCON*, 2003.

[31] Juha Latvala, Jari Syrjarinne, Hannu Ikonen, and Jarkko Niittylahti. Evaluation of rssi-based human tracking. In *Signal Processing Conference*, 2000.

[32] Willy Hereman and William Murphy. Determination of a position in three dimensions using trilateration and approximate distances. 1995.

[33] IOP-Institute of Physics. How does gps work? http://www.physics.org/article-questions.asp?id=55, last visited on April 2016.

[34] Karatsinides Spiro. Enhancing filter robustness in cascaded gps-ins integrations. *IEEE Transactions on Aerospace and Electronic Systems*, 30, 1994.

[35] Umut Orguner. Lecture notes in linear system theory 1, February 2013.

[36] Wikipedia. Newton's method — Wikipedia, the free encyclopedia, 2016. [Online; accessed 25-February-2016].

[37] Wikipedia. Bellman-ford algorithm — Wikipedia, the free encyclopedia, 2016. [Online; accessed 28-February-2016].

[38] XSens Technologies. Datasheet mti-1 series, 2016. [Online; accessed 28-February-2016].

[39] Rahni et al. 2d translation from a 6-dof mems imu's orientation for freehand 3d ultrasound scanning. *IFMBE Proceedings*, 21:699–702, 2008.

[40] Wikipedia. Methods of contour integration — Wikipedia, the free encyclopedia, 2016. [Online; accessed 10-February-2016].

[41] James Stewart. *Calculus: Early Transcendentals*. Cengage Learning, 2007.

[42] Mark Berg, Otfried Cheong, Kreveld. Marc, and Marc Overmars. *Computational Geometry*. Springer, 1998.

[43] Jop Frederik Sibeyn. Graph algorithms. https://www8.cs.umu.se/ jopsi/d-inf504/chap14.shtml, last visited on April 2016.

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Çimenci, Kadir
Nationality: Turkish (TC)
Date and Place of Birth: 11.07.1988, Ankara
Marital Status: Single
Phone: 0 533 7229934

EDUCATION

Degree	Institution	Year of Graduation
B.S.	Electrical&Electronics Engineering Faculty, ITU	2012
B.S.	Mechanical Engineering Faculty, ITU	2012
High School	Işıklar Askeri Lisesi	2006

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
2	Turkish Aerospace Industry, TAI	Design Engineer
2	HAVELSAN	Embedded Software Engineer