DYNAMIC FORMATION CONTROL WITH HETEROGENOUS MOBILE ROBOTS

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

BY

KADIR ÇIMENCI

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 HETEROGENOUS MOBILE ROBOTS

submitted by **KADIR ÇIMENCI** 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. xxx
Dean, Graduate School of **Natural and Applied Sciences** _____

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

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

Assist. Prof. Dr. xx
Co-supervisor, **Electrical and Electronics Engineering Department, METU.** _____

Examining Committee Members:

Prof. Dr. İsmail Hakkı Toroslu
Computer Engineering Department, METU _____

Prof. Dr. Ali H. Doğru
Computer Engineering Department, METU _____

Assoc. Prof. Dr. Pınar Karagöz
Computer Engineering Department, METU _____

Assist. Prof. Dr. Selim Temizer
Computer Engineering Department, METU _____

Assist. Prof. Dr. Aykut Erdem
Computer Engineering Department, Hacettepe University _____

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: KADIR ÇIMENCI

Signature :

ABSTRACT

DYNAMIC FORMATION CONTROL WITH HETEROGENOUS MOBILE ROBOTS

Çimenci, Kadir
M.S., Department of Electrical and Electronics Engineering
Supervisor : Prof. Dr. Aydan Erkmen
Co-Supervisor : Assist. Prof. Dr. xx

June 2016, 98 pages

Formation control in robotics is a growing topic where mainly research works are geared towards heterogenous swarm colonies under decentralized control or heterogenous colonies where some centralization is considered. In swarm works where decentralization is applied, it is nevertheless assumed 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 our thesis work 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 sensory capabilities under decentralized dynamically changing formation control. These robots are able furthermore, to just posess local mutual interactions only with their closeby neighboring agents. Using communications with those neighbors, all agents are being able to acquire graph theoretically, information about the whole colony. Simulations in our work will be generated using the Gazebo environment modeling a rough territory. Hardware applications of our approach will also be developed.

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

ÖZ

HETEROJEN ROBOTLARLA DİNAMİK FORMASYON KONTROLÜ

Çimenci, Kadir
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Aydan Erkmen
Ortak Tez Yöneticisi : Yrd. Doç. Dr. xx

Haziran 2016 , 98 sayfa

Formasyon kontrolü robotik alanında heterojen robot kolonilerinin merkezi yönetici birimler olmadan ya da yerel merkezi birimleri barındıran topolojilerle kontrolü konularına 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 basit 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 hedefliyoruz. 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 alacağız. Çalışma kapsamında ele alınan kolonilerdeki tüm robotlar yalnızca kendilerine en yakın komşu üyelerle yerel etkileşimlerde bulunabileceklerdir. Komşularıyla etkileşimde bulunan üyeler, konsensus koordinat sistemi ve kolonin geri kalanı hakkında bilgi sahibi olabileceklerdir. Çalışmamız kapsamında simülasyonlar Gazebo ortamında üç boyutlu düzgün olmayan araziler modellenerek yapılacaktır. Ayrıca donanımsal gerçeklemeler içeren çalışmalar da yapılacaktır.

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

To my family and people who are reading this page

ACKNOWLEDGMENTS

TODO

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ALGORITHMS

ALGORITHMS

LIST OF ABBREVIATIONS

ABBRV	Abbreviation

CHAPTER 1

INTRODUCTION

In this thesis work, it is aimed to create a novel method for formation control of a swarm which consists of heterogenous mobile robots. The term of swarm represents a large group of locally interacting individuals with common goals[4].

Self organizing swarm researches and its applications are generally inspired by the biological systems in the nature. The behaviours of these biological systems were considered mysterious and strange for a long time, but recent researches show that individuals don't need any sophisticated knowledge or top level functionalities to produce such complex tasks[2] . These biological systems (e.g. colony of ants) have simple behaviours but they can accomplish very complicated collective tasks in the nature which are impossible with their own individual capabilities. Beni [1] describes this collaboration of members as follows:

The group of robots is not just a group. It has some special characteristics, which are found in swarms of insects, that is, decentralised control, lack of synchronisation, simple and (quasi) identical members.

It is obvious that such a collective behaviour of these swarms has more power and efficiency than the sum of the individual capabilities of the members.

General aspects of the swarm robotics systems are the simplicity of individuals, restricted sensing and communication capabilities, achieving tasks mutually, robustness and decentralized control capability[6].

Swarm robotics has been studied to produce different collective behaviors 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[7] are studied on controllers for aggregation of coordinated motion of the identical mobile robots called swarm-bots. Hou, S.P., C.C. Cheah, and J.J.E. Slotine is focused on controlling of a swarm within a dynamically changing formation[8]. Ganesh and Lisa introduced two new strategies for collective search and exploration of fields with swarm intelligence[9]. 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[10]. Campo and Gutierrez is studied on collective foraging task and they propose a method for path selection to optimize the profits of the swarm[11].

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.

1.1 Problem Definition

In this thesis work, the main idea is to propose a complete design solution to a dynamically changing formation system, including a local positioning system and formation control system. The swarm which is used in this formation system is assumed to be composed of heterogenous agents which have different functionalities and physical properties. The formation shape is expected to be closed contours which cannot be identified with analytical expressions and these shapes will be changing dynamically with a continuous manner.

Formation control system is heavily depend on the position data of individual agents in the environment. 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. Even if a positioning solution depending on different methods including visual feedback(by image processing), RSS(received signal strength) etc. is available for an indoor application, 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. As a result of these constraints, it is required to implement a localization solution for the agents to provide the corrected positions in the workspace to be used by the formation control system. This localization process have to correct the position data of the agents with the determined process period and within an maximum error bound which will be determined by the requirements of the formation control problem.

Formation control system have to provide a solution to the coverage of a dynamically changing formation shape by the agents in the swarm. Desired formation shapes will not be simple geometrical shapes like circles, triangles etc. Heterogenous agents with different shapes have to cover the instant formation shape homogenously as possible as. The total displacements of the agents while travelling towards the desired formation have to be minimized. In other words, agents have to adapt the new formation shape with minimum possible movement. On the other hand, collision avoidance with the obstacles in the workspace and with the other agents has to be provided while achieving different types of formation tasks. Each agent must execute its own controller-decision making algorithms in a decentralized manner.

1.2 Motivation

The formation control problem can be defined as collaboration of a group of agent to maintain a formation with a certain shape [12]. It focuses on leading the individual agents of a swarm to perform collective tasks including shape generation and formation reconfiguration while traversing a trajectory by providing collision avoidance simultaneosuly. 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 homoge-

nous or heterogenous agents according to the requirements of the problem. The usage of the homogenous agents increases the total energy 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 heterogeonus agents holds the 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 heterogenous 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. 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 foot-bots are responsible for operations with their manipulators and eye-bots are responsible of observations and reconnaissance on the area.

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

A swarm which is composed with same type of homogenous 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 continous coverage and improved capability. They also mentioned that small formations will reduce the fuel consumption for propulsion and expand the sensing capabilities of microsatellites[15].

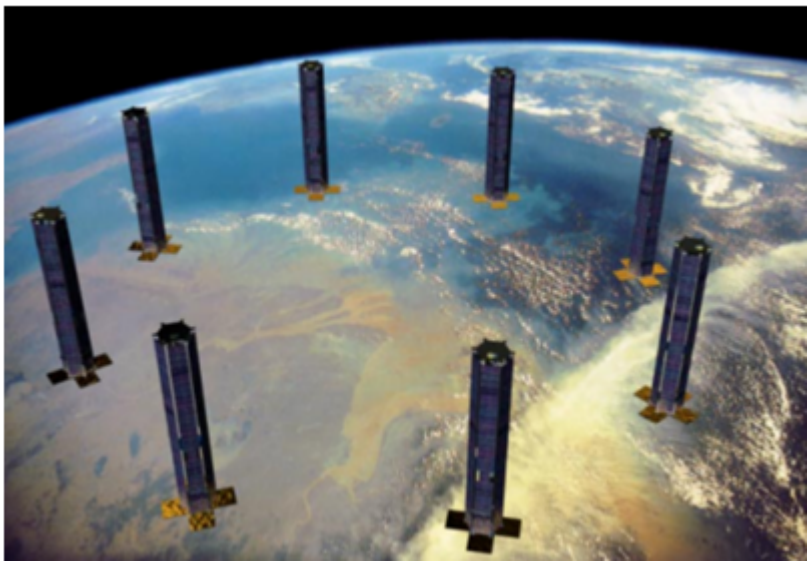


Figure 1.2: Sparse Aperture Formation of Micro Satellites

Formation control solutions has lots of usage areas such as coverage missions, security patrols and search&rescue in hazardous environments etc.[13]. For missions related with area coverage and reconnaissance, a group of autonomous vehicles may be required to keep in a specified formation [13]. Balch and Arkin presented a behaviour based formation control for multi robot teams which is implemented on a team of robotic scout vehicles manufactured for a DARPA project[14].

Figure 1.3: A Team of Four Robotic Scout Vehicles on which Formation Control Techniques Implemented

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.

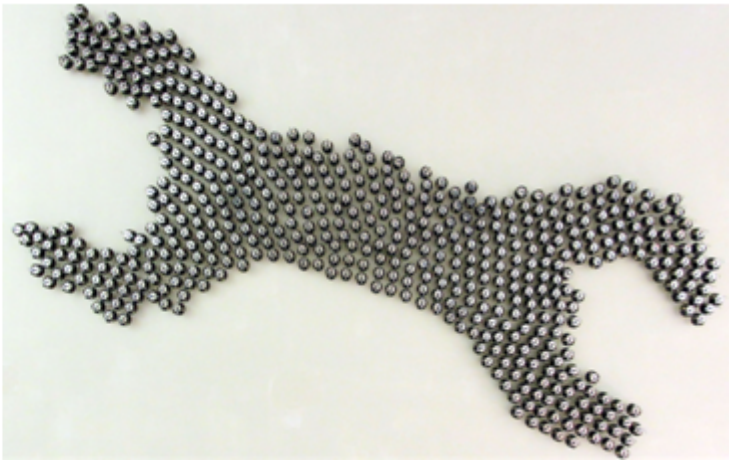


Figure 1.4: Formation Control with Kilobots

These micro robots have a great reusability for different types of formation control problems and they have biological insprations from the nature in the sense of individual simplicity and power of collective behaviors.

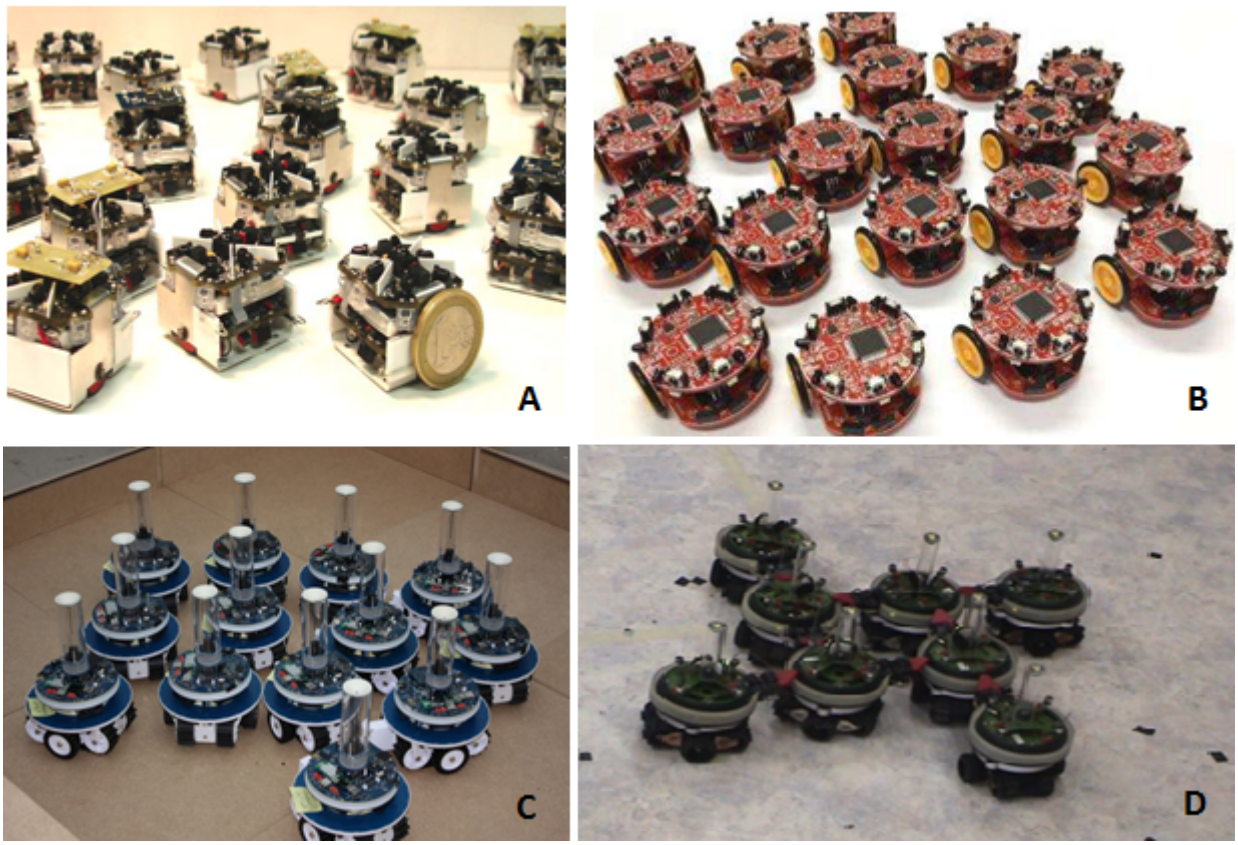


Figure 1.5: A) Swarm Robot Project from Universities of Stuttgart
B) Colias Project from University of Lincoln and Tsinghua University in China
C) Marx bot developed at EPFL
D)Swarm bots project conducted by European Commission

1.3 Objectives

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 while providing a global solution to the formation control problem like the heterogeneity vs. homogeneity of the agents, communication structures, centralized vs. decentralized structures, swarm control strategies like behavior based and leader-following approaches or virtual structure based approaches.

In addition to choice of the formation control infrastructures, capabilities of the in-

dividual 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. 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 colony of ants etc.

In this project the defined requirements and objectives are given as follows;

1.3.1 Heterogenous Robots with Different Dynamics

Agents have different dynamics from each other like different friction surfaces, geometrical structures and functionalities. They have different volumes and masses(not mass point particles) and they may collide with the other ones and the obstacles in the environment.

Figure 1.6: Heterogenous Agents with Different Physical Properties and Functionalities

1.3.2 Communication Infrastructure

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.

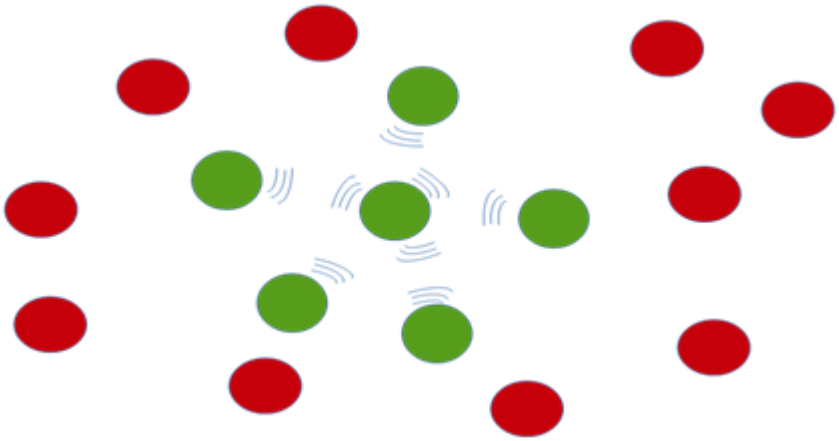


Figure 1.7: Radio Links on Agents Have a Narrow LOS Range

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 propogated along a route by transporting over the nodes(member agents of the swarm).

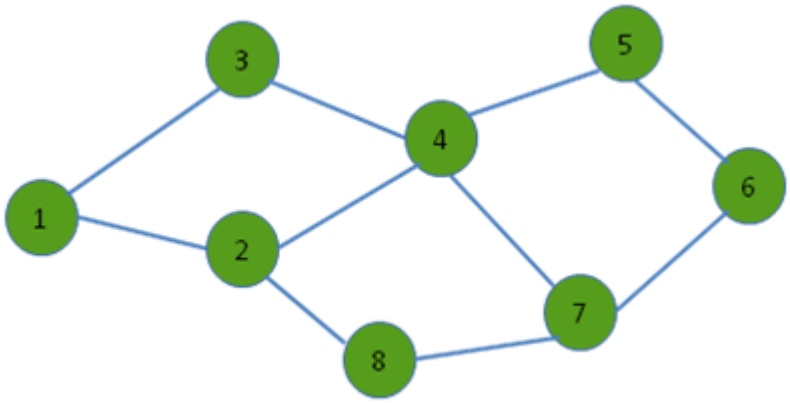


Figure 1.8: Mesh Network Between Agents

b)Communication Bandwith Bandwith 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 control objectives. This type of systems achieve superior performance and optimal decisions but they require high computational power, high communication bandwiths and are not robust due to dependence on a single controller[12]. 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 to reach as unaware of the other agents' choices.

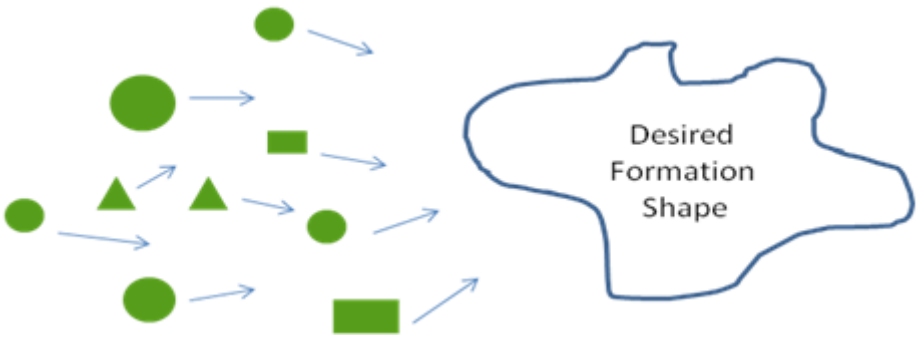


Figure 1.9: Agents make their own choices about target goal states

1.3.4 Complex Closed Contours

Formation shapes will be defined as closed curves with complex shapes and they cannot be identified analytically . On the other hand shapes will be changing dynamically during formation control.

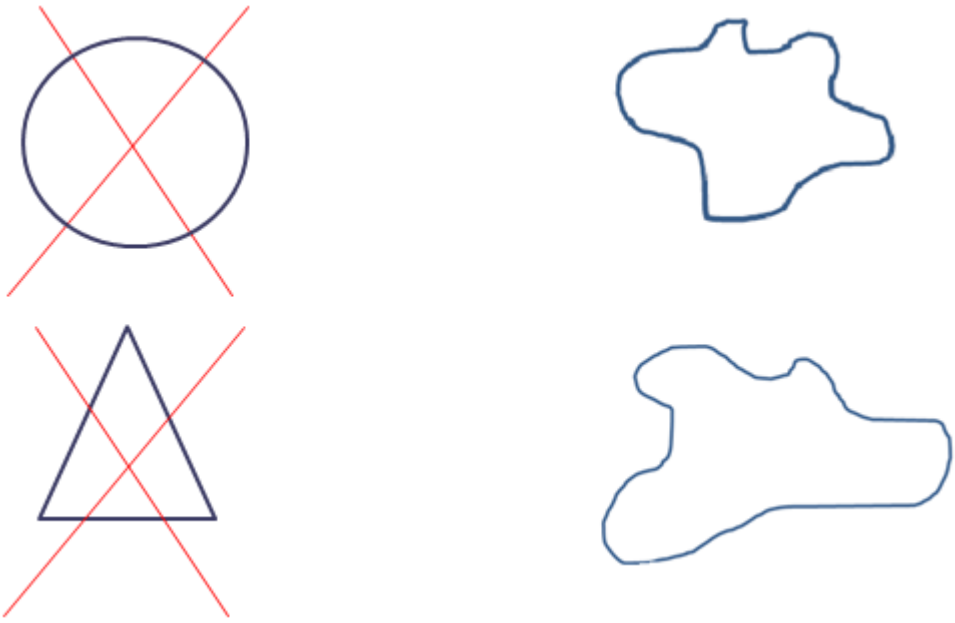


Figure 1.10: Complex and Dynamically Changing Formation Shapes

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. This condition must be taken into account during the control system design, since individuals do not have a high resolution and sensitive data about their state vectors, and they cannot execute high level complex control algorithms.

1.4 Goals

The objectives and the assumptions about the requirements define the goals of this thesis project. The goals of the project are listed in the following section.

1. Agents have to propogate their position and velocity states with the help of inertial measurements.This process is handled with a Kalman estimator which takes the translational accelaration 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 the position data with the help of agents which have

positioning sensors(position beacons) by local trilaterations.This position data is used in the internal estimator systems as external measurements to correct the drifts caused by propogation 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&bandwith radio links, the propogation 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. Differnt 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 the 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 propogate their position and velocity states with their inertial measurements. Due to the bias and drift errors on this solution a position update&adjust process is handled on 0.33Hz frequency

with the help of position beacons which are agents with position sensors on board in the swarm. During the update phase of the solution route tables for individual agents are determined with the help of Graph Theory based 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 clusters after the establishment of route tables. A Kalman estimator system is designed to fuse these propogation 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 methods and randomized fractals method. Desired complex formation shapes are partitioned into goal states according to the heterogenous agents in the swarm for both of these two methods. Decision process of the agents about their target goal states to optimize the overall utility 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 a high accuracy position data to the agents which do not have a 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. Partitioning the complex formation shapes into goal states to cover the whole

formation homogenously with the different types of heterogenous agents.

4. Designing and implement the rules&algorithms for the decision process of the individual agents about the goal states to reach.

5. Designing a simulation environment to test the proposals and algorithms of this thesis work

6. Designing a simple demonstrative hardware application.

1.7 Outline of the Thesis

This thesis work is organized into 5 main sections .Chapter 1 introduces the main theme and the potential areas of the usage 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 basic mathematical background on the methods used in this paper.

Chapter 3 introduces the methods and solutions used in two different parts of the problem; local positioning system and formation conroller system. In this chapter, routing algorithms and mathematical aspects of the trilateration process is introduced. 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

This chapter focuses on the related works on local positioning systems and formation control systems.

2.1 Local Positioning Systems

Positioning systems are used to provide the required position data&feedback to the systems where it is desired to control the location of the mobile agent in the workspace. These systems fall in to 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. 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[19]. A local positioning system can provide a position information where GPS systems are unavaliable,with the usage of signaling beacons which are placed at the exacltly known locations.

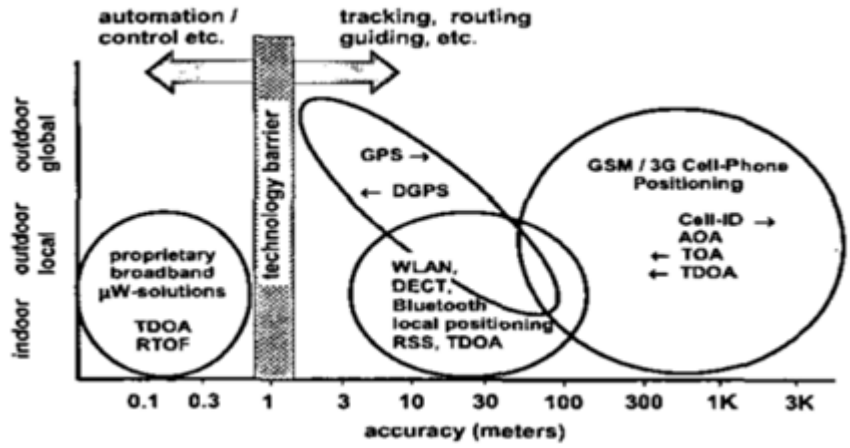


Figure 2.1: Accuracy Statistics of Different Positioning Sources

Figure xx represents an onverview of current positioning systems. Global positioning systems are widely used nowadays and they provide accuracies in the range of 3-30 meters, they can operate outdoor environments with the necessity of 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, but they can perform in indoor environments partially. Local positioning systems have the capability of working indoor environments and they have a wide accuracy range changing with respect to the implemented topologies and methods.

Local positioning systems has different system topologies illustrated in the Table xx[20].

Table 2.1: Local Posititioning Systems with Different System Topologies

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[20]. In self positioning system a mobile device finds its own position with the help of a reference like 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 with respect to its own position[19]. These two type of topologies can be converted to each other with the help of a 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.

2.1.1 Measurement Principles

Angle of arrival (AOA), received signal strength(RSS) and propogation-time based systems are commonly used as three different measurement techniques used in local positioning systems.

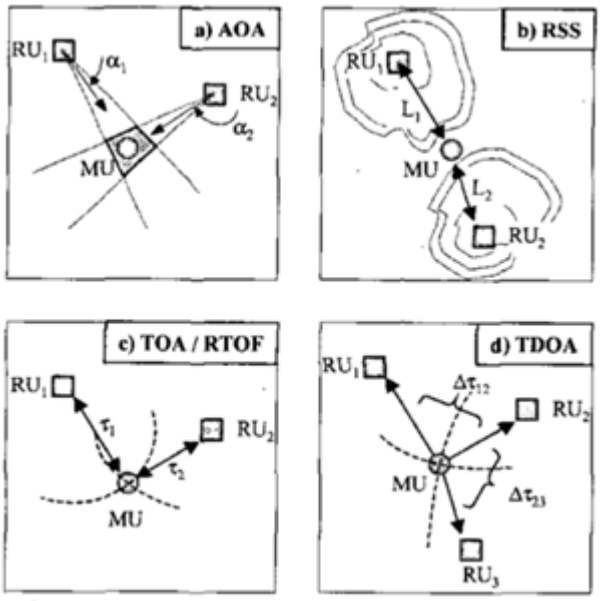


Figure 2.2: Different Measurement Principles

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

Received signal strength(RSS) systems calculate the distance value by taking the difference of the received signal power from the transmitted power. Some advanced propogation models are required to calculate the distance from the transmission loss in the air to eliminate the multipath fading and shadowing effects[21] .

Time based systems calculates the distance between measuring unit and signal transmitter with the help of propogation time like used in the global positioning systems generally. This process requires a perfect time synchronization between the mobile and stationary units[20].

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.

2.1.2 Trilateration Process

Trilateration process is used to determine the three dimensional position of unknown locations with the help of distance measurement to known positions [22]. 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 calculte 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 -xx demonstrates a simple trilateration procees 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.

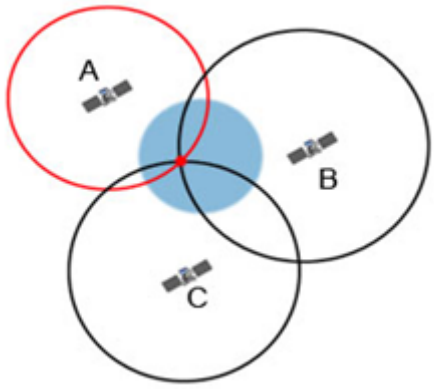


Figure 2.3: Trilateration Process

2.2 Formation Control Systems

Formation control problem have different subproblems like formation shape generation, formation reconfiguration&selection and formation tracking [12]. In formation

shape generation, agents are expected to get a formation shape which can be defined by externally or with some mathematical constraint functions[16]. One general approach is to consider some artificial potential functions. Samitha and Pubudu have presented an artificial potential function based method by considering the problem as controlling and positioning of a swarm into a shape bounded by a simple closed contour in the complex plane while spreading members inside the contour uniformly. They provide analysis about the stability and robustness of their systems with the help of Lyapunov like functions[17].

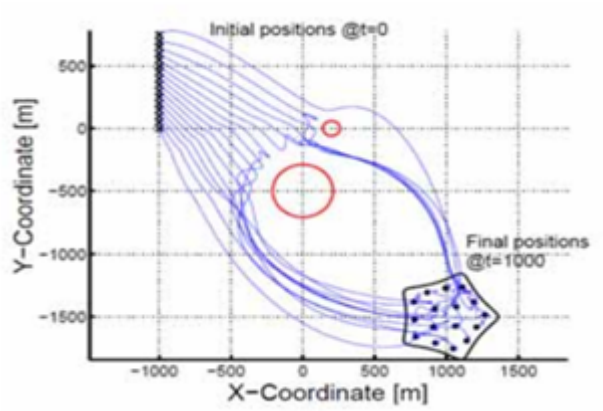


Figure 2.4: Motions and Formation of the Agents in Presence of Obstacles[17]

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. This rask requires formation reconfiguration and selection capabilities for the swarms. Hou and Slotine 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[8].

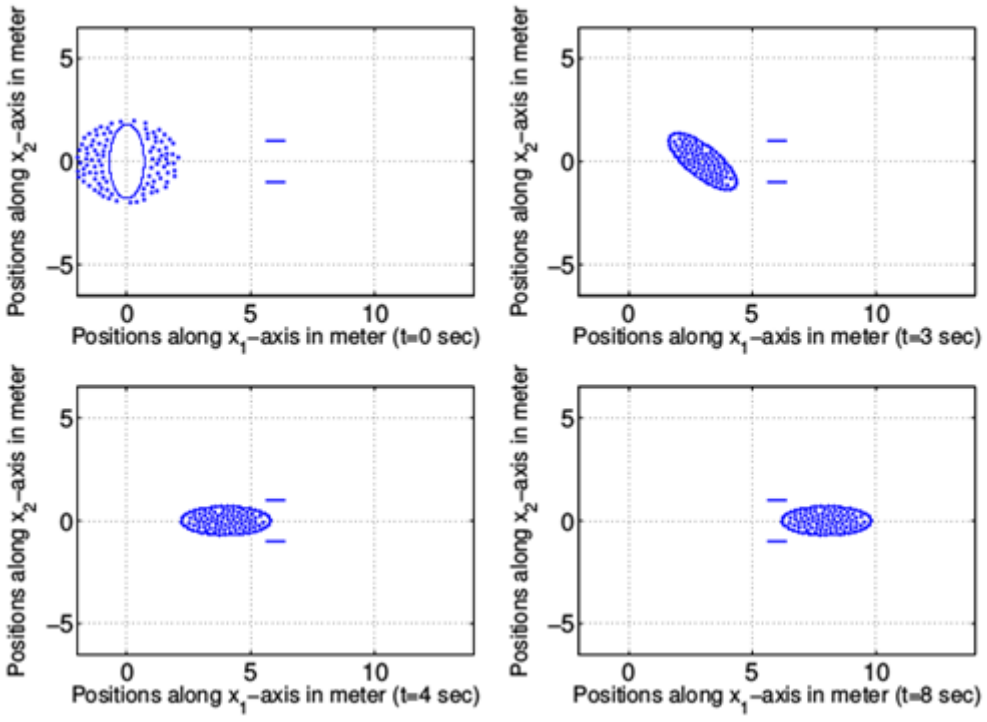


Figure 2.5: A Group of 100 Robots in a Rotating and Scaling Ellipse Formation[8]

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 most general strategy to provide a solution for this problem is leader-following swarm structures. Other strategies have a basis on optimization and graph theory approaches[12]. Kumar, Fierro and Das proposed a vision based formation control framework for this problem. This framework has a leader following background [18].

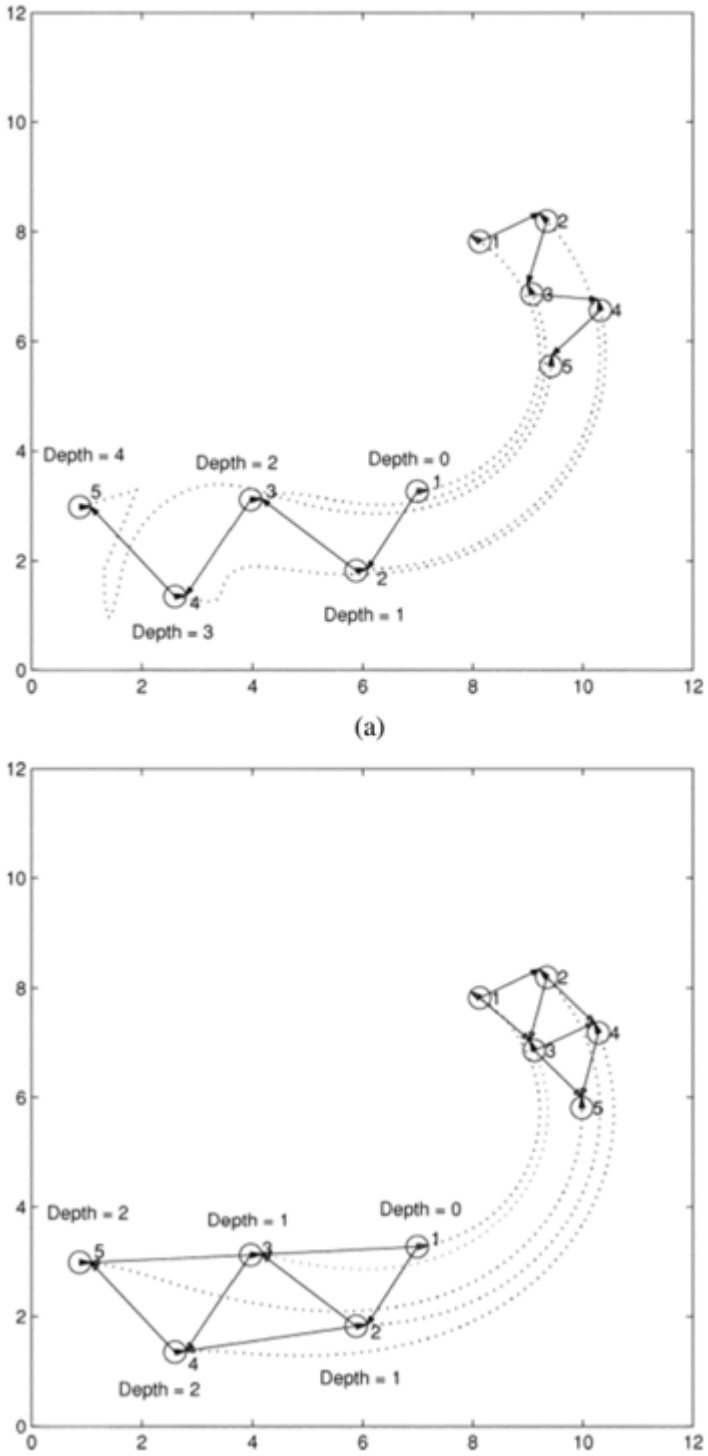


Figure 2.6: Five Robot Formation With Trajectory Tracking [18]

The solutions for the formation control approaches can be classified into three basic strategies as leader-following, virtual structure and behaviour based approaches[12]. 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. Kumar, Fierro .. at [18] proposed a control framework in which follower agents move along a trajectory afterwords the leader agent with a desired seperation l_{ij} and desired relative bearing angle ψ_{ij}. Figure -xx represents a formation control with three agents where R3 is the leader and R1,R2 are the follower agents.

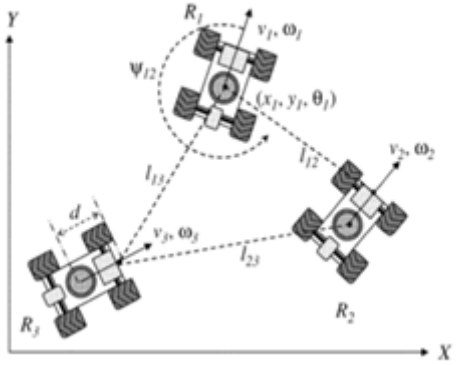


Figure 2.7: Leader-Follower Systems

In this approach it is hard to gather the agents in a certain shape. Another drawback is that, determining the seperation and bearing angles for individual agents will be getting harder with the increasing number of agents in the swarm 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 whole virtual structure and then the individual agent control laws are determined with inverse dynamic solutions[12]. Lewis and Tan proposed a virtual structure based method for formation control in [23] with a bidirectional flow control where robots move to stay in the virtual structure when the swarm is following a trajectory and virtual structure move to fit robots' current positions to compensate the relative errors at the end of that maneuver.

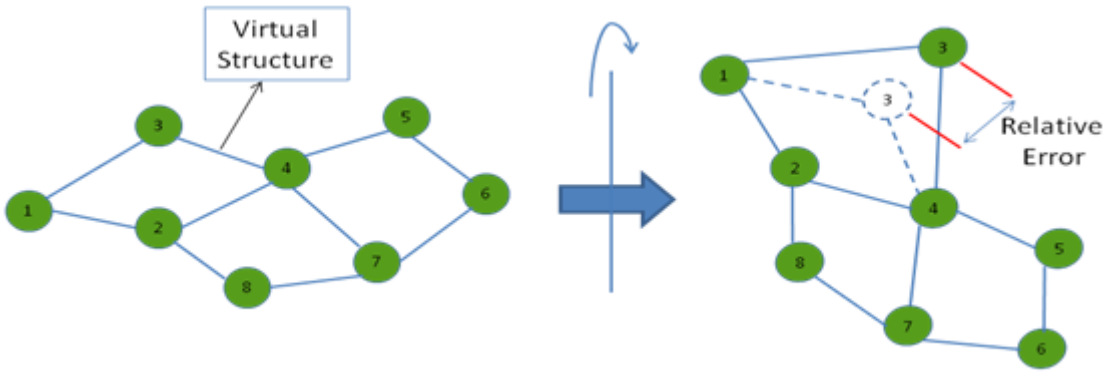


Figure 2.8: Rotational Maneuver of a Formation and Compensation of Virtual Structure

In virtual structure strategy it is easy to achieve a coordinated behavior 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 agents in the swarm.

Behavior based strategies model every agents' behaviors to achieve specific tasks with swarm. These behaviours may be very simple like randomly walking and avoiding obstacle in the environment or they may be defined very complicated to achieve complex formation shapes with the entire swarm while optimizing the overall energy consumption depend on 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[24]. In this approach, individual control laws for the agents are composed with the artificial forces defined between the agents (to avoid collisions) and between the desired formation shape. 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. 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 to optimize the total distance travelled by the agents. Another drawback, related with this type of solution, there is a possibility to have local min-

imas in the solution where an agent reaches an undesired configuration in a balance with different types of artificial force components. In this strategy the solution may converge to the steady state very slow due to absence of generalized goal states for individual agents in the final state of formation.

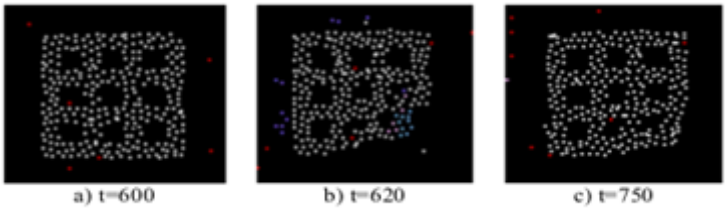


Figure 2.9: Formation Control with Artificial Forces

Another approach is to define mathematical constraints and objective functions to achieve a specific formation shape and controlling the swarm to follow a trajectory while keeping the formation. Kumar and Belta 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 seperately handled to manipulate the lie group G to achieve formation tracking and orientation control and to manipulate the shape S to achieve different geometrical shapes. Cheah and Slotine proposed a similar method based on objective functions[8]. Common drawback for these researches, they can only implement a limited number of simple geometrical shapes because the desired formation shapes must be identified analytically to compose the related objective functions or shape manifolds.

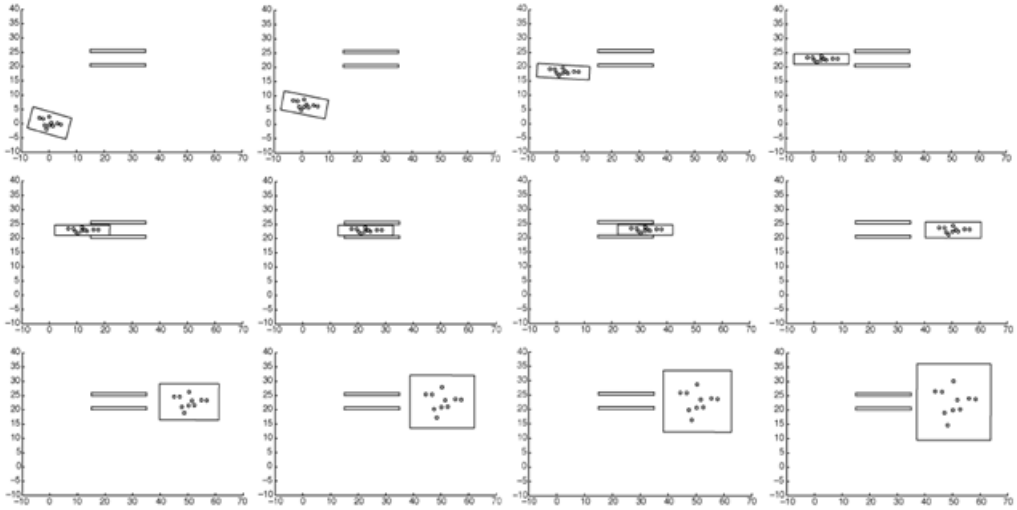


Figure 2.10: Formation Control with Objective Functions

2.3 Partitioning Complex Geometrical Shapes

This process is used to determine the goal states of the agents in the formation to cover the desired complex geometrical shape. There are some different solutions in the literature including fractal filling of space algorithms, bubble&circle packing algorithms and advancing front 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 filling space problems. Shier and Bourke [26] 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 statements.

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

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 known to converge for $c > 1$ and $N > 0$. In view of equation 2.2 one can write

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

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. Some of the outputs of their algorithm is given at Figure -xx

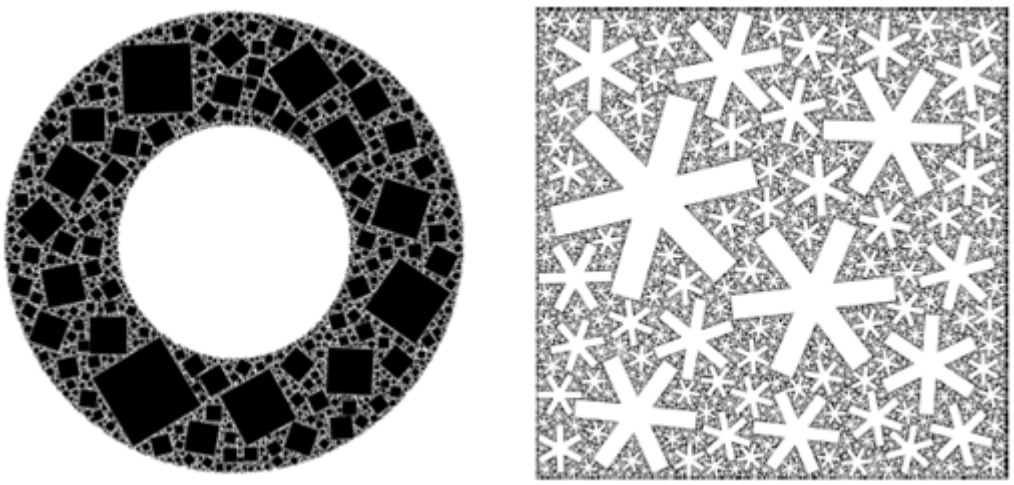


Figure 2.11: Space Filling Examples with Randomized Fractals

Bubble&Circle packing algorithms are widely used for mesh generation problems in finite element method. The main idea is that the close packing of bubbles mimics a pattern of Voroni tessellation. Corresponding to well shaped Delaunay triangles or tetrahedras which select the best topological connection for a set of nodes by avoiding small included angles[27].

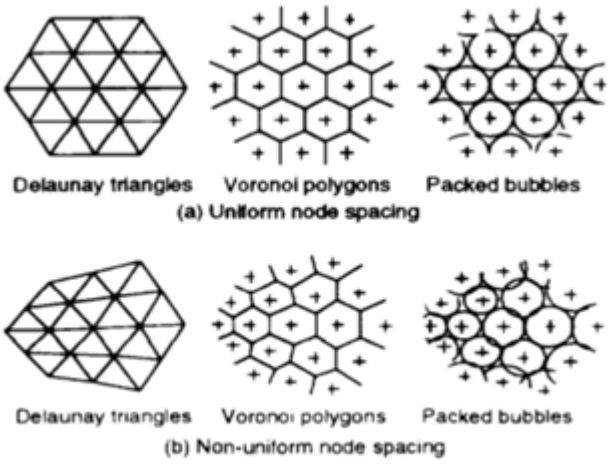


Figure 2.12: Uniform and Non-Uniform Node Spacing

Shimada and Gossard proposed a method based on interbubble forces to provide close packaging of bubbles in desired geometrical shape. This approach is very similar with the one used in formation control of swarms to achieve geometrical shapes. The related interbubble forces are described at Figure – xx.

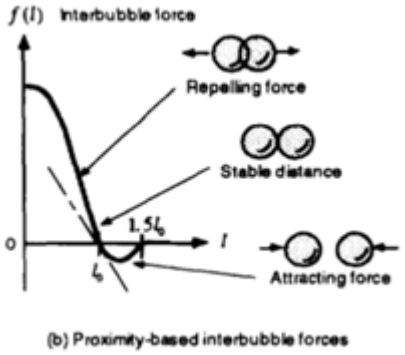


Figure 2.13: Interbubble Forces

With the help of adaptive population control by removing the excess bubble which significantly overlap their neighbors, they provide an adaptive bubble packing algorithm for mesh generation. A result with a 2D shape is given at Figure -xx

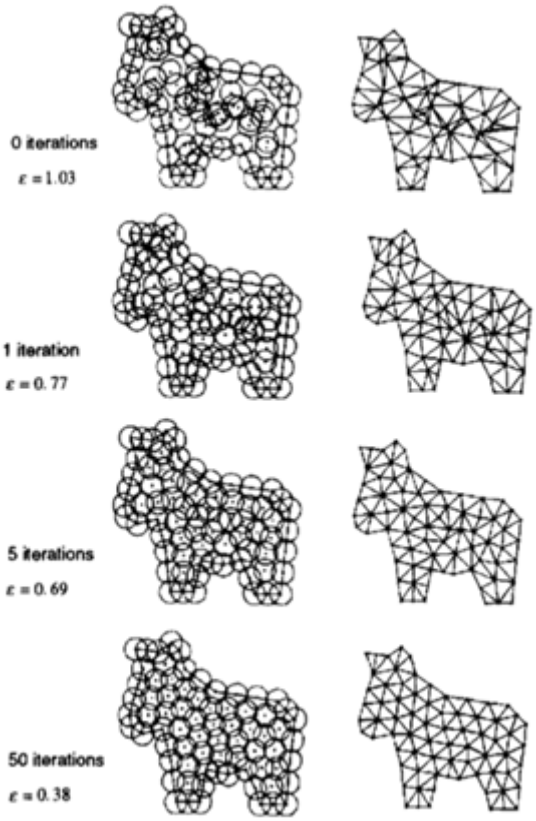


Figure 2.14: Mesh Generation with Interbubble Forces

This approach can easily be augmented for different types and number of shapes to partition a complex geometrical shape with regular sets. Basically the resultant solution will be similar to the one used in artificial potential field approach in formation control.

Advancing front methods are one of the alternatives used in mesh generation in literature. In a two dimensional advancing front method, new triangles are added into the domain from the initial front boundary and the front is propogated iteratively between the meshed and the unmeshed region. The initial front is created by the desired outer boundary of the shape and the procedure continues until the given domain is fully meshed.

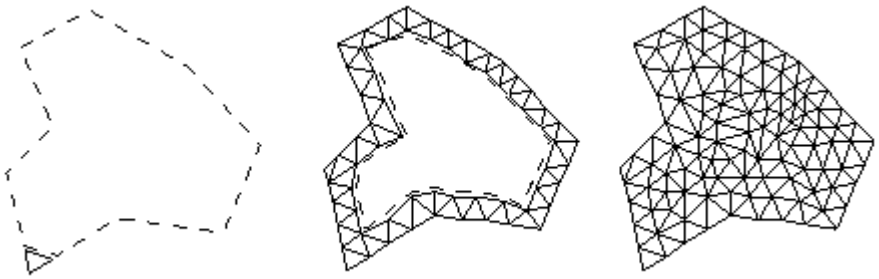


Figure 2.15: Triangulation with Advancing Front Method

CHAPTER 3

METHODOLOGY

In this thesis work, the problem of dynamical formation control of heterogenous mobile robots is reduced down to two subproblems as local positioning system design and formation control system design. Local positioning subsystem proposes a solution for the localization of the agents in the environment with the low sensor capabilities. This part of the solution provides a basis for the formation control problem with the true state vectors of agents composed with the translational positions and velocities. Formation control subsystem proposes a solution to achieve desired complex shapes with heterogenous agents in the swarm. In this chapter, details of the solution for these subproblems are presented in details.

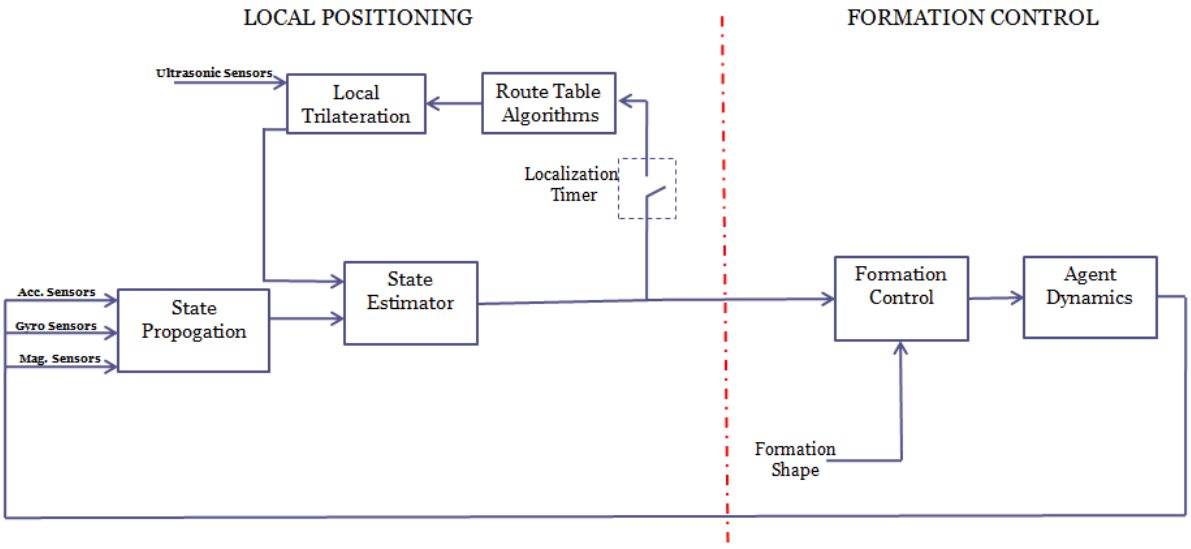


Figure 3.1: Flowchart of the Provided Solution

3.1 LOCAL POSITIONING SYSTEMS

Local positioning system is a subsystem to provide a complete solution to the localization of the agents in the environment. 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. The rest of the swarm have to maintain their position&velocity data with the help of their inertial measurements. Propogation of the states with these type of measurements are always inclined to drift problems due to the errors&noise and bias problems of the sensors. As a result of this problem, state vectors composed with the translational positions and velocities have to be corrected with an external measurement. A complete solution for this type of problem is proposed in Figure 3.2.

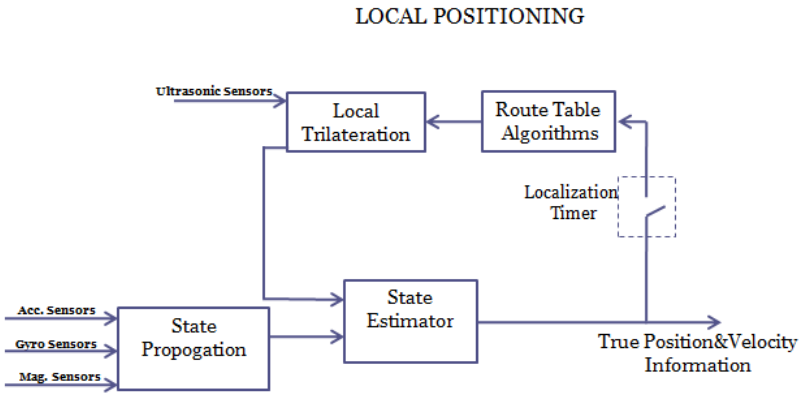


Figure 3.2: Local Positioning System

As illustrated in Figure 3.2, agents are expected to propogate their states with the help of acceleration, gyroscope and magnetometer sensors. Since the trilateration and route table determination processes require recursive and time consuming algorithms, a localization timer with a suitable period is implemented to the solution. Agents will correct their state vectors with the localization period of 3 seconds by measuring their positions with trilateration process and execute the update procedure in their state estimator algorithms. The requirement about the maximum value of 3 seconds for the localization period is determined with the help of Monte Carlo simulations discussed in Section-xx by defining a maximum allowable error on the position data since the error on the state vectors are drifting exponentially with increasing time.

3.1.1 Trilateration Process

Trilateration process helps the agents to localize themselves with the help of their local neighbors. Trilateration calculations use distance measurements to the nodes with known positions, to determine the coordinates of unknown positions[22]. These measurements are assumed to be done with the help of time of arrival methods disccussed in Section 2.1.1. Solution of the unknown position with the help of these distance measurements can be reduced down to a $A\vec{x} = \vec{b}$ type equation with the help of mathematical manipulations. The agents which has position sensors on their boards are called position beacons. Figure **??** illustrates a sample environment which consist of i beacons and an agent which tries to localize itself with the help of distance measurements r_i and the position data B_i of the beacon nodes.

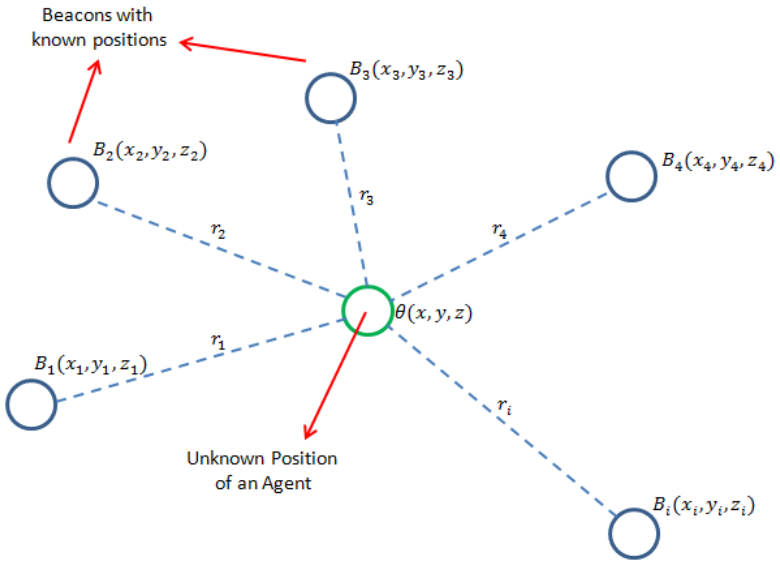


Figure 3.3: Environment for Trilateration Process

The distance function $\hat{r}_i$ can be written as follows;

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

where i denotes the beacon number and n is the total number of beacons. We have n number of constraints in the solution of the localization problem. In our work, we have implemented a two dimensional localization solution with the assumption of

each agent in the swarm have the same vertical position in Earth centered coordinate system. With this assumption, the problem for the localization process can be reduced down to a $A\vec{x} = \vec{b}$ type linear system problem and the constraints will be circle functions rather than spherical ones, presented with

$$(x - x_i)^2 + (y - y_i)^2 = {r_i}^2 \tag{3.2}$$

Lets assume $\theta = (x, y)$ is representing the coordinates of an agent which is trying to localize itself, 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 agents with exactly known positions.

If any of the 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 target agent and any beacon i

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

The distance between the referance beacon and the other beacons

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

The distance between the target agent and the referance beacon

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

Adding and subtracting x_j, y_j and z_j in equation 3.3 gives

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

This equation yields to

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

this general statement is valid for each beacon with

$$\begin{aligned}
&(x_2 - x_1)(x - x_1) + (y_2 - y_1)(y - y_1) = \frac{1}{2}[d_r^2(\theta) + d_{2r}^2 - d_2^2(\theta)] \\
&(x_3 - x_1)(x - x_1) + (y_3 - y_1)(y - y_1) = \frac{1}{2}[d_r^2(\theta) + d_{3r}^2 - d_3^2(\theta)] \\
&\ldots \\
&(x_n - x_1)(x - x_1) + (y_n - y_1)(y - y_1) = \frac{1}{2}[d_r^2(\theta) + d_{nr}^2 - d_n^2(\theta)]
\end{aligned} \tag{3.8}$$

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;

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

$$x = \begin{bmatrix} x - x_r \\ y - y_r \end{bmatrix} \tag{3.11}$$

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

with the help of these mathematical manipulations, localization problem is reduced down to a $A\vec{x} = \vec{b}$ problem.

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 world, the A matrix has $(n-1)$ rows and 2 columns, where n is the number of neighbor beacons. It is obvious that there is no solution when the number of neighbors lower than 3, since the A matrix will have one or less number of lines. When the number of neighbor 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 matrix has the dimensions of 2 x 2 and the rank of A matrix$(rank(A))$ is equal to 2, then the solution of $\vec{x}$ is unique with

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

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

2) Minimum Norm Solution With Pseudo Inverse:

If A matrix has the dimensions of $(n-1)$ x 2 where $n > 3$,which means the number of neighbor 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

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

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

with the help of the above equation

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

then

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

since A matrix is full column rank matrix,

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

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 $A\hat{x} = \vec{b}$ problem

3) Minimum Norm Solution With Nonlinear Least Squares 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,$(rank(A) = 1)$, then the solution to the $A\hat{x} = \vec{b}$ problem can be found iteratively with the help of nonlinear least squares method. Lets define the cost function to be minimized as the sum of the squares of the errors on the distances

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

with

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

There are various algorithms to minimize the sum of the square errors in literature, Newton iteration is used to find the optimal solution in this work. Taking the partial derivatives of the cost function with respect to x and y gives

$$\begin{aligned} \frac{\partial F}{\partial \vec{x}} &= 2\sum_{i=1}^{n} f_i \frac{\partial f_i(\theta)}{\partial x} \\ \frac{\partial F}{\partial \vec{y}} &= 2\sum_{i=1}^{n} f_i \frac{\partial f_i(\theta)}{\partial y} \end{aligned} \tag{3.22}$$

The partial derivative matrix of the cost function is composed as;

$$\bigtriangledown F(\theta) = 2 \begin{bmatrix} f_1 \frac{\partial f_1(\theta)}{\partial x} + f_2 \frac{\partial f_2(\theta)}{\partial x} + ... + f_n \frac{\partial f_n(\theta)}{\partial x} \\ f_1 \frac{\partial f_1(\theta)}{\partial y} + f_2 \frac{\partial f_2(\theta)}{\partial y} + ... + f_n \frac{\partial f_n(\theta)}{\partial y} \end{bmatrix} \tag{3.23}$$

Components of this partial derivative matrix converges to zero while the cost function iteratively optimized to a minimum point.

$$\bigtriangledown F(\theta) = 2J(\theta)^T f(\theta) = 0 \tag{3.24}$$

where

$$J(\theta) = \begin{bmatrix} \frac{\partial f_1(\theta)}{\partial x} & \frac{\partial f_1(\theta)}{\partial y} \\ \frac{\partial f_2(\theta)}{\partial x} & \frac{\partial f_2(\theta)}{\partial y} \\ ... & ... \\ \frac{\partial f_n(\theta)}{\partial x} & \frac{\partial f_n(\theta)}{\partial y} \end{bmatrix} \tag{3.25}$$

and

$$f(\theta) = \begin{bmatrix} f_1(\theta) \\ f_2(\theta) \\ ... \\ f_n(\theta) \end{bmatrix} \tag{3.26}$$

Using the vector $\vec{R}$

$$\vec{R} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{3.27}$$

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

$$\vec{R}_{\{k+1\}} = \vec{R}_{\{k\}} - (J_{\{k\}}^T J_{\{k\}})^{-1} J_{\{k\}}^T \vec{f}_{\{k\}} \tag{3.28}$$

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

$$J^T J = \begin{pmatrix} \sum_{i=1}^{n} \frac{(x-x_i)^2}{(f_i+r_i)^2} & \sum_{i=1}^{n} \frac{(x-x_i)(y-y_i)}{(f_i+r_i)^2} \\ \sum_{i=1}^{n} \frac{(x-x_i)(y-y_i)}{(f_i+r_i)^2} & \sum_{i=1}^{n} \frac{(y-y_i)^2}{(f_i+r_i)^2} \end{pmatrix} \tag{3.29}$$

and

$$J^T \vec{f} = \begin{pmatrix} \sum_{i=1}^{n} \frac{(x-x_i) f_i}{(f_i+r_i)} \\ \sum_{i=1}^{n} \frac{(y-y_i) f_i}{(f_i+r_i)} \end{pmatrix} \tag{3.30}$$

3.1.2 Route Table Determination and DSDV Algorithm

It is obvious that the possibility of having a large error on position and velocity data for the agents which do not have an external position measurement sensors,then it will be appropriate to get the agents into local trilateration process with the agents which have position sensors as much as possible. It is assumed that the positions of the beacon agents in the trilateration process are well-known with a little error boundary, ideally with no errors, so getting in the trilateration process with the agents which are already have errors on their position data and range measurements with errors may increase the error on the calculated datas. On the other hand due to the restrictions & requirements defined in the Section 1.3, it will not be possible to interact with the agents with position sensors directly due to the small communication ranges of the

agents and line of sight issues. In this case, it will be a good choice to handle this trilaterion process starting with the agents which are closer to the position agents in an increasing order of distance.

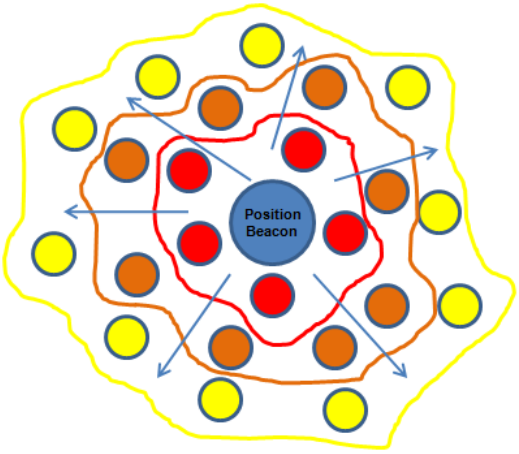


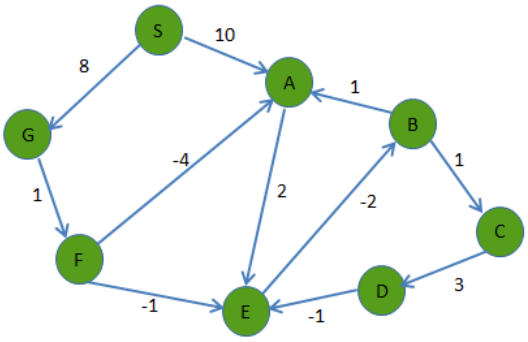
Figure 3.4: A Cluster of Agents Around a Position Beacon

As illustrated on the Figure 3.4 , suppose that the red agents which are closest ones to the position beacon in the swarm are the only ones which have the capability of interacting with the position beacon. Since the only source for true position measurement is the position beacon, this data must be distributed to these red agents first, then trilateration process must be propogated through the orange agents and then the yellow agents last, since they can only interact with an upper layer in the swarm due to the line of sight and communication range issues.

It is needed to have an algorithm to organize the order of the local trilateration process and to determine the beacon agents for each member of the swarm. Basically this algorithm will assign each agent a rank which represents the number of sequence in the localization process, and will determine at least three local neighbors of each agent to get in trilateration. Agents which do no have at least three neighbors are assumed to be lost agents and the handling of these type of agents are illustrated in Section 3.1.4.

3.1.2.1 Routing Algorithm- Bellman Ford

As mentioned in the Section 1.3 of the thesis work, agents are assumed to have a limited communication range and bandwith and the communication topology in the swarm must be implemented with a wireless mesh network. In this type of network, each node 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 bandwith data through the network with multiple hops. In this work, we have implemented this topology with a table driven routing scheme known as DSDV(Destination-Sequenced Distance Vector Routing Protocol) algorithm based on Bellman Ford algorithm. Bellman-Ford is and algorithm that computes the shortest path in a weighted graph. It is an algorithm based on relaxation, the correct distance to the vertices in the graph are updated iteratively from the initial estimations until converging to the optimal solution. This algorithm is slower than the Djikstra's algorithm which has similar functionalities but negative edge weights can be implemented in the related graph to report the negative cycles which means there is no cheapest path to the related destination vertex. On the other hand, it is possible to augment this algorithm with DSDV implementation to handle the routing loop problem when there is one or more vertices are no more exists in the network. The probability of the case with the non-existence of some vertices during the algorithm is processed can be very high since the agents in the swarm have low sensor capabilities and small range of communication and they have a great possibility to get lost in the environment. A simple demonstration of the Bellman-Ford algorithm with a simple network is illustrated in the Figure 3.5



	Iteration							
Node	0	1	2	3	4	5	6	7
S	0	0	0	0	0	0	0	0
A	∞	10	10	5	5	5	5	5
B	∞	∞	∞	10	6	5	5	5
C	∞	∞	∞	∞	11	7	6	6
D	∞	∞	∞	∞	∞	14	10	9
E	∞	∞	12	8	7	7	7	7
F	∞	∞	9	9	9	9	9	9
G	∞	∞	8	8	8	8	8	8

Figure 3.5: Costs for Shortest Paths to Each Nodes from Node 'S'

The algorithm to calculate the shortest paths for node 'S' terminates at the end of 7 iterations. At the beginning of the process, the weights for each edges are determined including the negative ones and each distance to the paths are filled with infinitive. Then the shortest paths to the each node in the given directed graph are determined iteratively with the help of the Bellman-Ford algorithm

3.1.2.2 Usage on Bellman Ford algorithm and DSDV

Bellman Ford algorithm have 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. Figure 3.6 illustrates a simple routing loop problem.

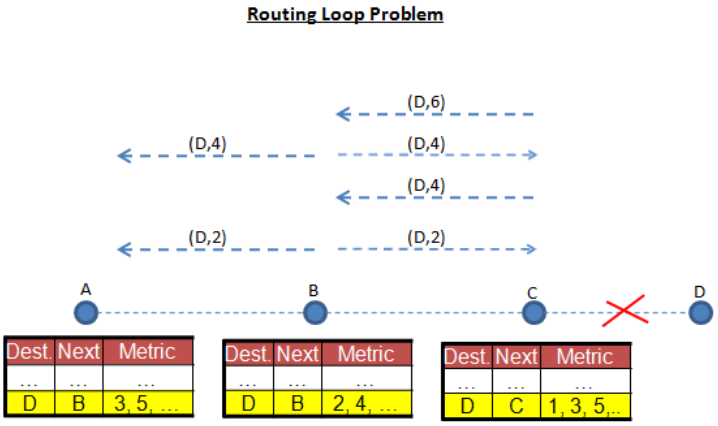


Figure 3.6: Routing Problem Engaged by a Lost of a Node in Network

Suppose that the node D have lost its contact with the network due to some malfunction or being lost by getting outside of the communication range to the closest neighbor of itself. Before this event, node C have a unit distance to the node D and consequently node B have a 2 unit distance to node D , node A have a 3 unit distance to node D. In case of a failure on node D, on the next iteration C will update its route table with the 3 unit distance to node D by taking reference the node B. Then node B will update its route table with the shortest distance of 4 units to the node D by referencing the node C and this process will diverge to infinity on the shortest paths with the increasing number of iterations. To provide a solution for these type of problems, DSDV algorihm has implement the sequence numbers and counts for hops into the route tables of the nodes. A simple route table for a vertex in a network is given in Figure 3.7

Route table for agent 2

Destination	Next Hop	Metric	Dest. Seq. No
1	1	1	123
3	3	1	516
4	4	1	212
5	4	2	168
6	8	3	372
7	8	2	432

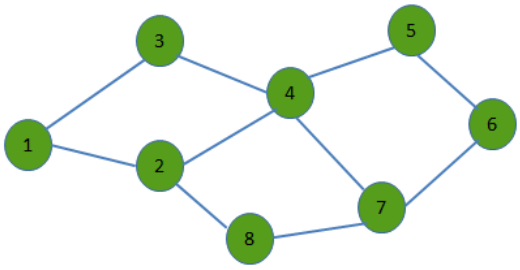


Figure 3.7: An Example for Route Table

In the DSDV algorithm, each node have a sequence number and counts for hops (metric) for each route in its route table and periodically transmits the updates including its own sequence number and routing tables updates. In the network, when two routes to the same destination received from two different neighbors, the nodes will observe the following rules;

- Choose the one 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.

3.1.2.3 DSDV Link addition

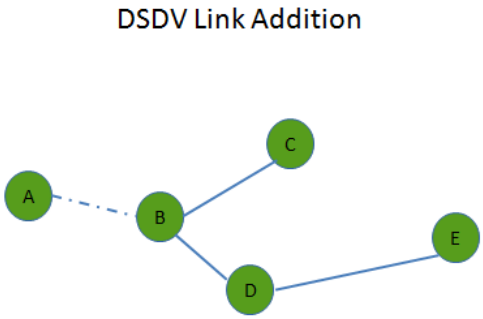


Figure 3.8: An Example for Route Table

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 propogates this new node to its neighbors

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

3.1.2.4 DSDV link breaks

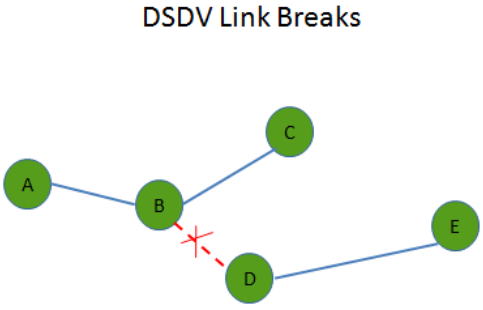


Figure 3.9: An Example for Route Table

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 hop count for node D and E to the infinity and increments the sequence numbers to these routes
- Node B propogates 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.

DSDV is implementing an algorithm to find the shortest paths between the internal nodes of a given directed graph. The costs for each shortest path are calculated with the help of the weight of edges in the graph. Since our aim is to find the closest position beacon which has the minimum number of hops, the weight for each edge in the graph must be represented with the same unit size and the directions of the edges are negligible.

3.1.3 Clusters

Since there are limited number of position beacons in a swarm, it will be appropriate to cluster the agents around these position agents to minimize the problem to the subproblems in which every one of them there are only one position beacon and the agents which are assigned to that cluster. The error on the trilateration process is expected to be increasing at the lower layers of the process illustrated in Figure -xx because of the cumulative effects of errors added to the position and velocity data of the agents in each layer. Thus, the policy for the assignment of the agents to the clusters must be the number of hops to the position beacons rather than the physical distances. Since DSDV algorithm has a structure storing the number of counts to each route in the tables, this information can be used to determine each agents' clusters in the swarm.

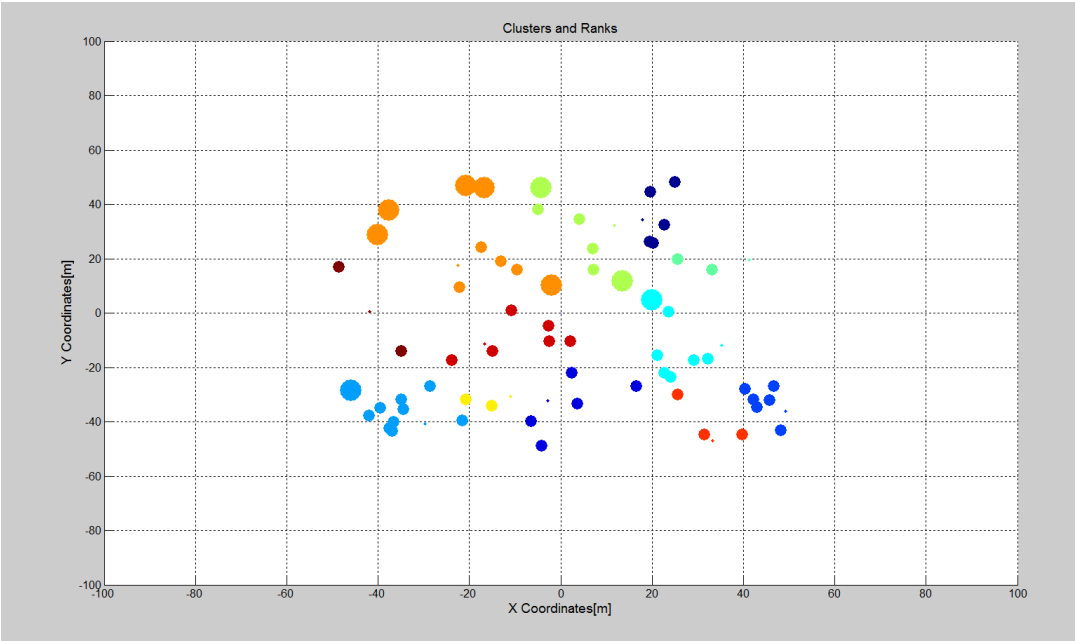


Figure 3.10: Clusters Around Position Beacons

As illustrated in the Figure 3.10, all agents have assigned themselves to the clusters around position beacons, in which they have minimum number of hops in the route to the related position agent. In the figure, the size of the agents are representing the increasing number of hops in the same cluster and the colors of the agents are representing the clusters in which each agent assigns itself. With this approach, the generic algorithm which will be executed with the period of localization process for each agent must be implemented as follows:

1) Update route table including the routes to the position agents in the swarm with DSDV algorithm
2) Check for the routes to the position agents and join the cluster in which minimum number of hops required to that destination.
3) Wait for the localization sequence in which the agents are entered the process with the increasing number of hops, e.g. the agent which have a single hop to the position agents are processed first, and then the other ones are processed consecutively as illustrated in Figure 3.4
4) If the localization sequence is valid for this agent, enter the trilateration process with at least 3 neighbor agents starting from upper layer, which are the next hops in the route table.

5) Call the update procedure of the observer system of which details are presented in section -xx

3.1.4 Handling Lost Agents

The minimum number of neighbors required for the trilateration process is three for a two dimensional localization problem as illustrated in Section 2.1.2 . Since the agents are assumed to have a narrow communication range, it is possible to not to find three neighbors for any agent at an instant time. In this case, it will be impossible to relocate these agents with trilateration and the position&velocity data will drift from the real values with the increasing time passed without trilaterions. To avoid these kind of problems, the 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 neighbors 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 the lost agent to get in touch with the rest of the swarm with directing it to the center of the swarm. A simple demonstration of this procedure is illustrated in Figure 3.11

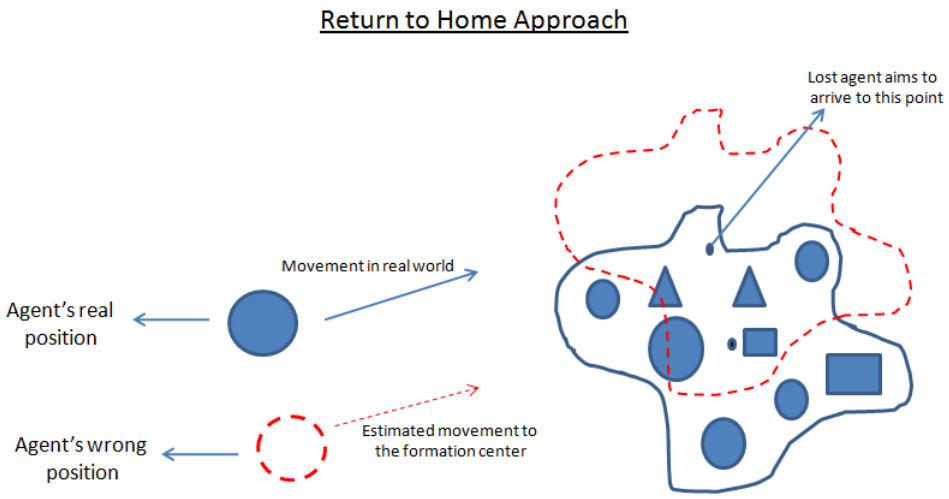


Figure 3.11: Return to Home Approach of a Lost Agent

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 other agents in the swarm even if it directs itself to an incorrect goal state.

3.1.5 State Estimation Procedure

In local positioning subsystem, agents are expected to execute a state estimator algorithm in which they propogate their state vectors composed of translational position and velocities with the help of inertial measurements. As discussed at the introduction of Section 3.1 they will update and correct their positions with the measurements provided by the trilateration process exeeuted with the localization timer period of 3 seconds. A Kalman estimator algorithm which uses the trilateration outputs as external measurents and the sensor measurement as inputs, is designed to fusion 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:

$$x_k = \begin{bmatrix} X_k \\ \dot{X}_k \end{bmatrix} \tag{3.31}$$

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 in y coordinates.

The linear model to propogate states will be:

$$x_{k+1} = F_k x_k + B_k u_k + w_k \tag{3.32}$$

where w_k is the process noise and

$$F = \begin{bmatrix} 1 & d_t \\ 0 & 1 \end{bmatrix} \tag{3.33}$$

$$B = \begin{bmatrix} \frac{{d_t}^2}{2} \\ d_t \end{bmatrix} \tag{3.34}$$

where d_t is the propogation period and u_k is the translational acceleration calculated with the help of AHRS system driven by inertial sensors. The observation(external measurement) which will be calculated with the trilateration process :

$$z_k = H_k x_k + v_k \tag{3.35}$$

where v_k is the measurement noise. Since the trilateration process will provide new position informations of the agents:

$$H_k = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{3.36}$$

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

$$w_k = \mathcal{N}(\mathbf{0}, \mathbf{Q_k}) \tag{3.37}$$

$$v_k = \mathcal{N}(\mathbf{0}, \mathbf{R_k}) \tag{3.38}$$

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 covariance of R_k

The filter has two main subsections named predict and update phases. The update phase of the filter is executed after each trilateration process with a period of 3 seconds. The filter equations are as follows:

Propogation phase:

$$\hat{x}_{k,k-1} = F_k \hat{x}_{k-1,k-1} + B_k u_k \tag{3.39}$$

$$P_{k,k-1} = F_k P_{k-1,k-1} F_k^T + Q_k \tag{3.40}$$

Update Phase:

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

$$S_k = H_k P_{k,k-1} H_k^T + R_k \tag{3.42}$$

$$K_k = P_{k,k-1} H_k^T S_k^{-1} \tag{3.43}$$

$$\hat{x}_{k,k} = \hat{x}_{k,k-1} + K_k \tilde{y}_k \tag{3.44}$$

$$P_{k,k} = (I - K_k H_k) P_{k,k-1} \tag{3.45}$$

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

$$Q_k = \begin{bmatrix} Max.\,Acceleration\,Error * \frac{d_t^2}{2} & 0 \\ 0 & Max.\,Acceleration\,Error * d_t \end{bmatrix} \tag{3.46}$$

$$R_k = Max.\,Position\,Error\,on\,Trilateration\,Process \tag{3.47}$$

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 heterogenous mobile robots is presented in this chapter. Basically three different approaches as artificial forces method, bubble packing method and randomized fractals method are implemented. 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 homogenously 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. The other two methods , shape partitioning based approaches, share a common structural basis. In these approaches the complex formation shape is partitioned into goal states to cover the shape homogenously 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.

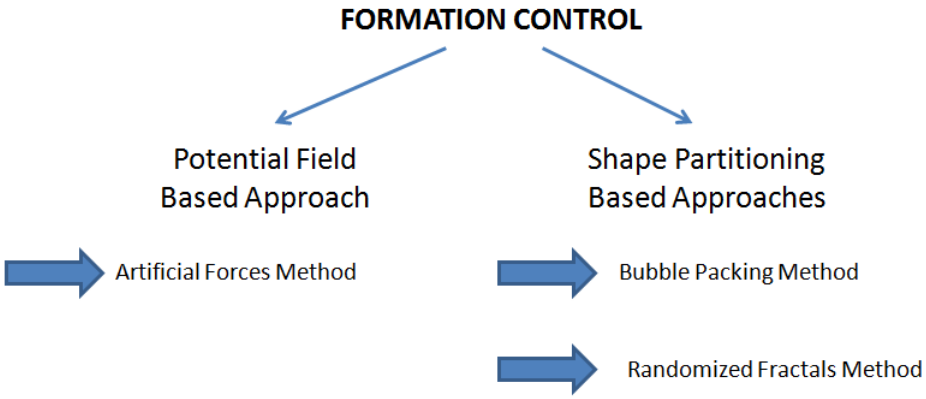


Figure 3.12: Formation Control Topologies

3.2.1 Potential Field Based Approach

3.2.1.1 Artificial Forces Method

Artificial forces method implements some potential fields over each agent arised from the interactions between agents, formation shape and environment etc. 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, e.g. 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

$$x = x(t),\ y = y(t),\ a \leq t \leq b \tag{3.48}$$

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

$$z(t) = x(t) + iy(t),\ a \leq t \leq b \tag{3.49}$$

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 mentioned in Equation 3.50

$$z(t_1) \neq z(t_2)\ whenever\quad t_1 \neq t_2 \tag{3.50}$$

On the other hand if $z(a) = z(b)$ is the only intersection point, the curve is said to be simple closed curve. Regarding with these given definitions, an example for a simple smooth closed curve is illustrated in Figure 3.13

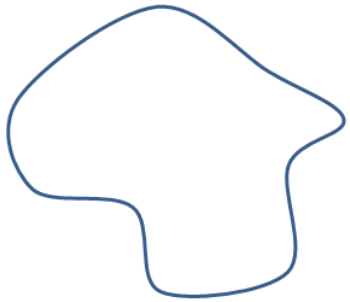


Figure 3.13: A Simple Closed Curve

Let $f(z)$ is a complex function in a domain D in the complex pane 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

$$\oint_C f(z)dz = \int_a^b f(z(t))\frac{dz(t)}{dt}dt \tag{3.51}$$

where

$$\frac{dz(t)}{dt} = \frac{dx(t)}{dt} + i\frac{dy(t)}{dt},\ a \leq t \leq b \tag{3.52}$$

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

$$\begin{aligned}\oint_C f(z)dz &= \oint_C udx - vdy + i\oint_C udy + vdx \\ &= \int_a^b \left[u(x(t),y(t))\frac{dx(t)}{dt} - v(x(t),y(t))\frac{dy(t)}{dt}\right]dt \\ &\quad + i\int_a^b \left[u(x(t),y(t))\frac{dy(t)}{dt} + v(x(t),y(t))\frac{dx(t)}{dt}\right]dt\end{aligned} \tag{3.53}$$

This simplified form of the contour integrals will be used in utility function in the following section.

Utitilty Functions

As it is mentioned in the Section 1.3 part, the formation shapes will be simple closed contours which cannot be identified analytically, but the definition for the artificial forces and utility functions are expressed in continuous contour integrals which requires the analytical expression of the curve on which the integral will be taken. To provide a solution to these type of calculations it is required to redefine these statements in discrete domain to achieve calculations with closed curves without analytical expressions.

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 counterclockwise around the point. This number is used to switch on/off some of the artificial forces while the agent is inside or outside of the formation shape. 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 Caucy winding number is :

$$n(C,z_i) = \frac{1}{2\pi i}\oint_C \frac{dz}{z - z_i} \tag{3.54}$$

The winding number for agent i in the swarm,

$$n(C,z_i) = \begin{cases} 1 & \text{when member i is inside C} \\ 0 & \text{when member i is outside C} \end{cases} \tag{3.55}$$

To implement this statement in discrete domain

$$n(C,z_i) = \frac{1}{2\pi i}\oint_C f(z)dz \tag{3.56}$$

where

$$f(z) = \frac{1}{z-z_i} \tag{3.57}$$

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

$$u(x,y) = real(f(z)) \;\; and \;\; v(x,y) = imag(f(z)) \tag{3.58}$$

partitioning as it mentioned in Equation 3.53

$$\oint_C f(z)dz = \oint_C udx - vdy + i\oint_C udy + vdx \tag{3.59}$$

then

$$n(C,z_i) = \frac{1}{2\pi i}\left[\int_a^b \left(u\frac{dx}{dt} - v\frac{dy}{dt}\right)dt + i\int_a^b \left(u\frac{dy}{dt} + v\frac{dx}{dt}\right)dt\right] \tag{3.60}$$

Discrete contour integral representation of this equation is

$$n(C,z_i) = \frac{1}{2\pi i}\left[\sum_{k=1}^{K}\left(u(x_{k+1}-x_k) - v(y_{k+1}-yx_k)\right) + i\sum_{k=1}^{K}\left(u(y_{k+1}-y_k) + v(x_{k+1}-x_k)\right)\right] \tag{3.61}$$

where

$$\|z_k - z_{k-1}\| = \|z_{k+1} - z_k\|, \;\; \forall k; \;\; when \;\; K \to \infty \tag{3.62}$$

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 distances. This approach is used to provide representations of the contour integrals in discrete domain.

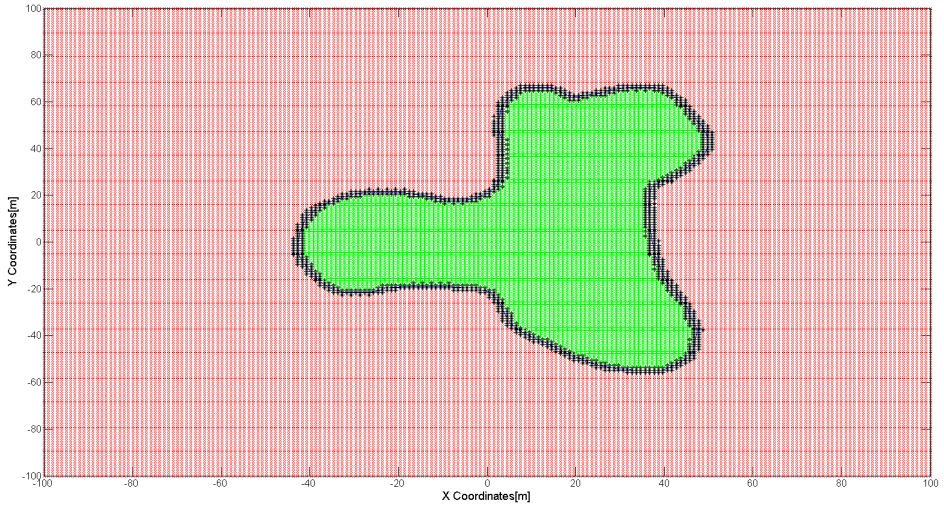


Figure 3.14: Formation Shape in an Environment (Green Dots: Inside of the Shape - Red Dots: Outside of the Shape)

2- Length of a formation shape

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

$$l(C) = \oint_C \|dz\| \tag{3.63}$$

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

$$l(C) = \sum_{k=1}^{K} \sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2} \tag{3.64}$$

where

$$\|z_k - z_{k-1}\| = \|z_{k+1} - z_k\|, \ \ \forall k; \ \ when \ \ K \to \infty \tag{3.65}$$

3-Center of a Formation Shape

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

$$z_c = \frac{\oint_C z \|dz\|}{l(C)} \tag{3.66}$$

the discrete domain expression for Equation 3.66 with points of $z_k = (x_k, y_k)$ in the complex plane

$$\begin{aligned} z_{cx} &= \frac{\sum_{k=1}^{K} x(k)}{K} \\ z_{cy} &= \frac{\sum_{k=1}^{K} y(k)}{K} \end{aligned} \tag{3.67}$$

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

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

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

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

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

the discrete domain expression for Equation 3.71 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

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

Artificial Forces

Artificial forces are defined to gather the agents in the swarm inside a formation shape and make them distributed homogenously 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. Attractive forces, repulsive forces, intermember forces, obstacle forces and transition forces are implemented to generate individual control laws for all agents in the swarm. 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 [xx]

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

where

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

with

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

The vector for individual control laws of the swarm

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

where

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

It is necessary to define 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 3.15 below

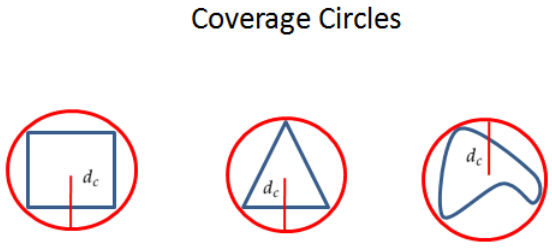


Figure 3.15: Coverage Circles of Different Types of Agents

The components of these control forces 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.

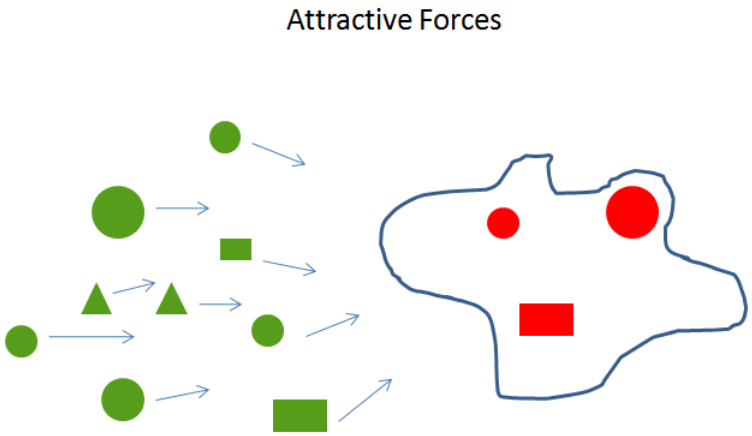


Figure 3.16: Attractive Forces Generated by the Formation Shape

The equations for the attractive forces are defined as follows:

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

where k_a is the variable gain for the attractive forces. 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:

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

where

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

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.

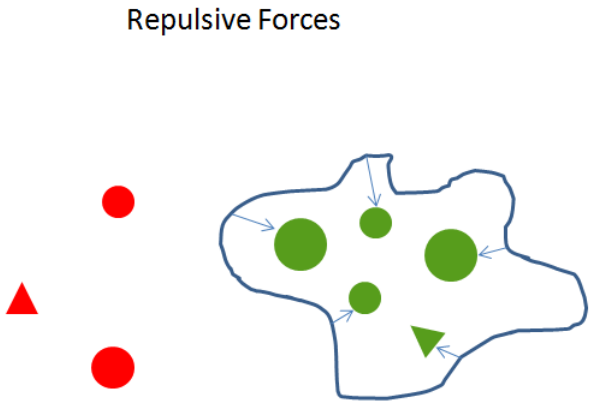


Figure 3.17: Repulsive Forces Generated by the Formation Shape

The equations for the attractive forces are defined as follows:

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

where k_r is the variable gain for the repulsive forces. 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:

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

where

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

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.

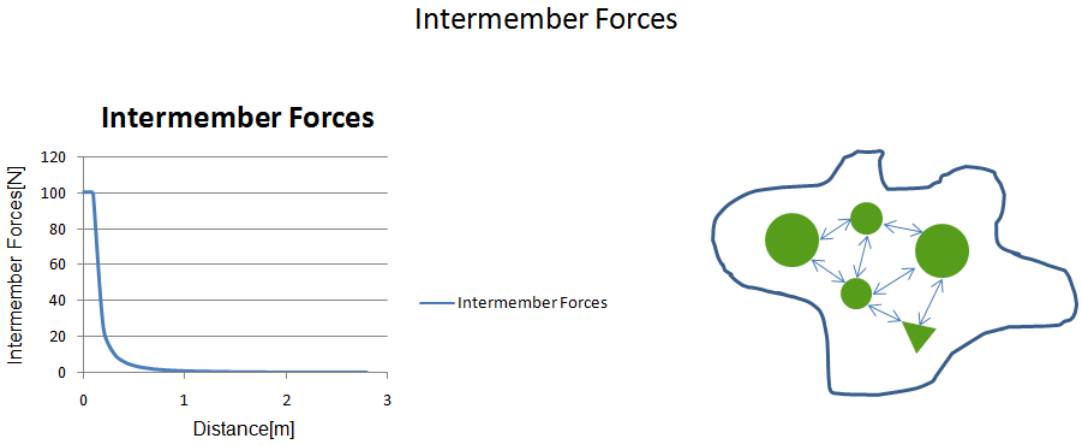


Figure 3.18: Intermember Forces Generated by Agents

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

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

where d_o is the total distance minimum distance between the agents which can be calculated with

$$d_o = d_i + d_j \tag{3.87}$$

The force components in x,y coordinates respectively,

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

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, it is needed 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.

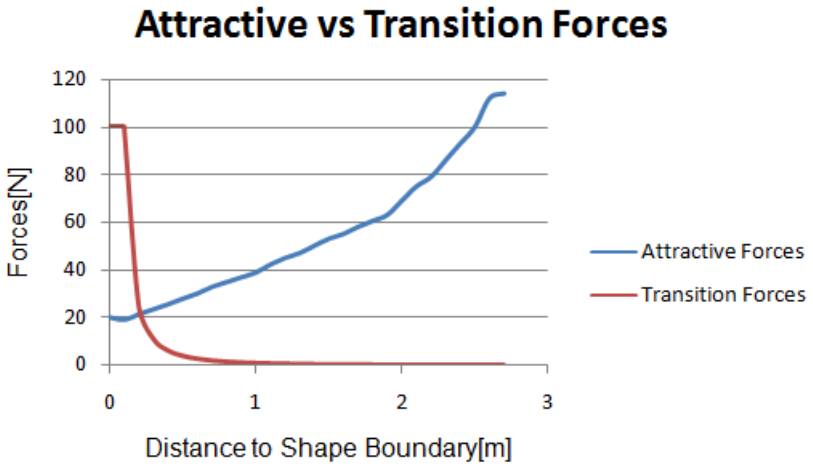


Figure 3.19: Comparison of Attractive and Transition Forces Close to the Formation Shape Boundary

The equation for the transition forces are defined as follows:

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

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.

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:

$$F_{i,o} := k_o \oint_O \left[\frac{\alpha X_i - z_o}{(\|\alpha X_i - z_o\| - d_{c_i})^4} \right] \|dz_o\| \tag{3.91}$$

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.

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:

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

where

$$\|z_{ok} - z_{ok-1}\| = \|z_{ok+1} - z_{ok}\|, \ \ \forall k; \ \ when \ \ K \to \infty \tag{3.92}$$

Buffer Zone Implementation

The attractive and transition forces are defined to be active when the agents are outside of the shape, the resistive forces are active when agents are inside the shape. Due to this type of 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, a buffer zone around the boundaries are implemented. The main approach during the transition of these boundaries is to dynamically change the variable gains of these artificial forces to linearly change between zero and nominal values at the boundary conditions.

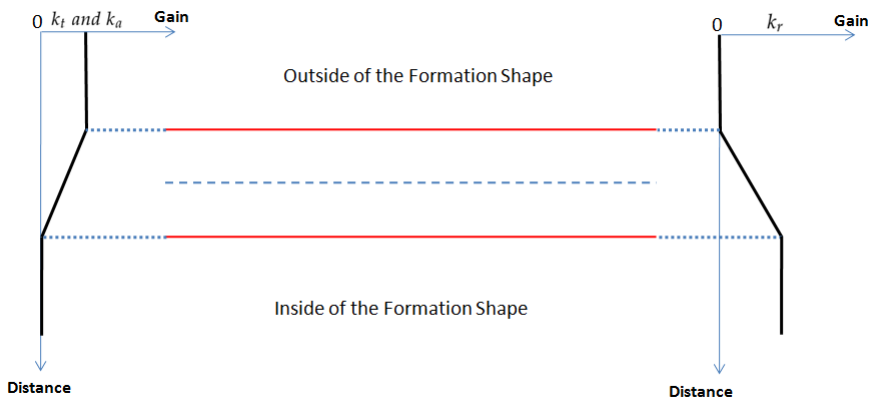


Figure 3.20: Transition of the Artificial Forces on Buffer Zone

XSwarm Theory

In artificial forces method it is required to analyse the movement of a single agent and to prove that the motion will be towards to the desired shape under the effect of different artificial force components, since there are no predetermined goal states within the desired formation shape. For this purpose, Samitha and Pubudu [xx] have provided a definition of "X-Swarm" and give a theorem which describes the motion of an agent towards the formation shape based on the X -Swarm definition. Let S be a swarm with N agents. This swarm is defined as X-Swarm if there exists some positive constants δ and γ that satisfy the following conditions simultaneously for all $i, j \in S$ and $i \neq j$

$$\begin{aligned} &\succ \quad d_{ij} \geq \gamma + \delta \\ &\succ \quad \left\| \frac{z_i - z_{cm}^i}{z_i - z_{cm}} \right\| < \left(1 + \frac{\delta}{\gamma}\right)^3 \end{aligned} \tag{3.93}$$

where

$$d_{ij} = \left\| z_i - z_j \right\| and z_{cm}^i = \frac{\sum_{j=1; j \neq i}^{N} z_j}{N-1} \tag{3.94}$$

The term of z_{cm}^i is the center of mass of the swarm without the i^{th} member.

Lemma 1: The magnitude of artificial inter-member force acting on an agent of X-

Swarm is less than

$$\frac{k_m(N-1)}{\gamma^3}\left\|z_i - z_{cm}\right\| \tag{3.95}$$

Proof:

$$F_{i,m} = k_m \sum_{j=1, j\neq i}^{N} \frac{z_i - z_j}{d_{ij}^3} \tag{3.96}$$

Using the first condition of the X-Swarm , we have

$$\left\|F_{i,m}\right\| < \frac{k_m(N-1)}{(\gamma+\delta)^3}\left\|z_i - z_{cm}^i\right\| \tag{3.97}$$

Then using the condition 2 of the X-swarm definition, the following expression can be derived

$$\left\|F_{i,m}\right\| < \frac{k_m(N-1)}{\gamma^3}\left\|z_i - z_{cm}^i\right\| \tag{3.98}$$

This result provides an upper bound to the magnitude of the inter-member repulsion force in a X-Swarm.

Theorem: Consider the agent i outside of a desired formation shape in a X-swarm, if $\frac{k_a}{k_m} > \frac{N-1}{\gamma^3 l(C)}$, then the motion of member i is towards to the center of the swarm.
Proof: Choosing a Lyapunov function candidate for the agent i as:

$$V_i = \frac{1}{2}\dot{v}_i\dot{v}_i^T + \frac{1}{2}v_i v_i^T\left(k_a l(C) - \frac{k_m(N-1)}{\gamma^3}\right) \tag{3.99}$$

It is possible to show that $\dot{V}_i$ is bounded by taking the derivatives,

$$\dot{V}_i \leq -k_f\left\|\dot{v}^2\right\| + \begin{bmatrix} \left\| k_m \sum_{j=1, j\neq i}^{N} \frac{z_i - z_j}{\left\|z_i - z_j\right\|^3} \right\| \\ -\frac{k_m(N-1)}{\gamma^3}\left\|v\right\| \end{bmatrix} \left\|\dot{v}\right\| \tag{3.100}$$

Since agent i is considered a member of a X-swarm, from Lemma 1

$$\left\| \sum_{j=1, j\neq i}^{N} \frac{z_i - z_j}{\left\| z_i - z_j \right\|^3} \right\| < \frac{N-1}{\gamma^3} \left\| z_i - z_{cm} \right\| \tag{3.101}$$

and hence

$$\dot{V}_i < -k_f \left\| \dot{v}_i \right\|^2 \tag{3.102}$$

which proves the given theorem.

Implementation of X-Swarm Theory

Two conditions to create a X-Swarm is to find two positive variables which satisfy the following inequalities

$$\begin{aligned} &\succ \quad d_{ij} \geq \gamma + \delta \\ &\succ \quad \left\| \frac{z_i - z_{cm}^i}{z_i - z_{cm}} \right\| < \left(1 + \frac{\delta}{\gamma} \right)^3 \end{aligned} \tag{3.103}$$

It is possible to check the X-swarm conditions for a swarm by using $min(d_{ij})$ and $max \left\| \frac{z_i - z_{cm}^i}{z_i - z_{cm}} \right\|$ values. Lets define the variable of $X_{sw_{min}}$ and $X_{sw_{max}}$ as following:

$$X_{sw_{max}} := max \left\| \frac{z_i - z_{cm}^i}{z_i - z_{cm}} \right\| \quad X_{sw_{min}} := min(d_{ij}) \tag{3.104}$$

To satisfy the X-swarm inequalities, it is possible to write

$$X_{sw_{min}} = \gamma + \delta \;\; X_{sw_{max}} = (1 + \frac{\delta}{\gamma})^3 \tag{3.105}$$

by using the above equations it is possible to calculate the variables of γ and δ as following,

$$\gamma = \frac{X_{sw_{min}}}{\sqrt[3]{X_{sw_{max}}}} \tag{3.106}$$

and

$$\delta = X_{sw_{min}} - \frac{X_{sw_{min}}}{\sqrt[3]{X_{sw_{max}}}} \tag{3.107}$$

if the calculated γ and δ are both positive, than the X-swarm conditions are satisfied. Otherwise, the main approach is to increase the variable gain of k_m on runtime which determines the magnitude of the repulsive forces between agents to increase the minimum distance between the agents in the swarm, $X_{sw_{min}} := min(d_{ij})$

On the other hand, the theorem related with the convergence of member agents of an X-Swarm to the z_{cm} is satisfied under the condition of $\frac{k_a}{k_m} > \frac{N-1}{\gamma^3 l(C)}$. Another approach is to increase the k_a gain on runtime to make the related inequality be satisfied to force the agents which are outside of the desired shape to the center mass of the formation geometry.

3.2.2 Shape Partitioning Methods

Shape partitioning methods have basically 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 homogenously. There are two different solutions to the shape partitioning problem which are presented in this thesis work, bubble packing method and randomized fractals method. The second part of the solution is the decision process of the agents to select these goal states continuously to minimize the total displacements of the agents while achieving the desired formation shape. During this decision process, the cost of reaching different goal states will be the main criteria for each agent. It is obvious that each agent will try to choose a goal state with minimum cost according to its position and the orientation in the environment. But the cases in which two or more agents are willing to reach the same goal point must be handled to optimize the overall utility of the swarm.

Shape partitioning methods have an approach to direct the agents to the goal states in which they have assigned instantly to minimize the total displacement, but it is important to keep the agents together in the swarm while moving on a trajectory towards the formation shape, to reduce the lost agent events described in Section

3.1.4. On the other hand, it is required to provide a collision avoidance between the agents and the obstacles while reaching the desired goal states. For these purposes, attractive forces which directs the agents to the center of the formation shape under X-swarm consideration and inter-member forces which avoids collisions between agents are added to the agents' control signal as an additive term.

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 and tetrahedra can be created by connecting the centers of the bubbles[27]. The algorithm places the bubbles with their initial conditions in the surface and apply them interbubble forces which imitates the Van der Waals forces between the moleculer bonds to distrubute the bubbles homogenously. 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.

Since the numbers and the the radius of coverage circles which is defined at Section 3.2.1.1 for the agents are predetermined in our formation control problem, the general bubble meshing algorithm have to be adopted to meet the requirements for shape partitioning in formation control. The basic approach is to represent the agents in the swarm as bubbles with the radius of their coverage circles and create a mesh by using these bubbles.

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, the bubbles are initiated close to the center of the formation shape randomly. The algorithm for initial bubble place-

ments 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 shape 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. An execution of the Algorithm 1 is illustrated in Figure 3.21 below.

Figure 3.21: Initialization of the Bubble Packing Algorithm

II- Bubble Meshing Process

The bubbles are distributed homogenously 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

$$f_i(l) = \begin{cases} al^3 + bl^2 + cl + d & \text{when } 0 \leq 1 \leq l_0 \\ 0 & 1 > l_0 \end{cases} \tag{3.108}$$

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

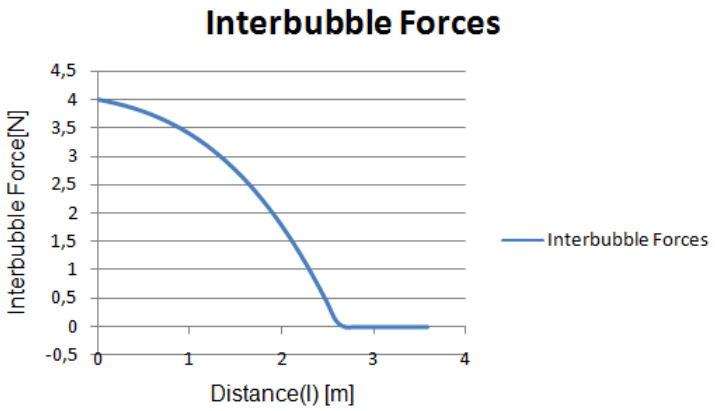


Figure 3.22: Interbubble Forces

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::

$$f_r(X_i) := \oint_C \left[\frac{\alpha X_i - z}{\|\alpha X_i - z\|^3} \right] \|dz\| \tag{3.109}$$

where k_r is the variable gain for the repulsive forces. 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:

$$f_r(X_i) = \sum_{k=1}^{K} \frac{Z_i - Z_k}{\|\alpha X_i - z_k\|^3} \tag{3.110}$$

where

$$\|z_k - z_{k-1}\| = \|z_{k+1} - z_k\|, \ \ \forall k; \ \ when \ \ K \to \infty \tag{3.111}$$

The bubbles are distributed homogenously under the influence of these two forces when they get an equilibrium state in which the total net forces acting on individual bubbles reaches zero. The final equilibrium states of the bubbles determines the potential goal states of the agents in the swarm to cover the formation shape.

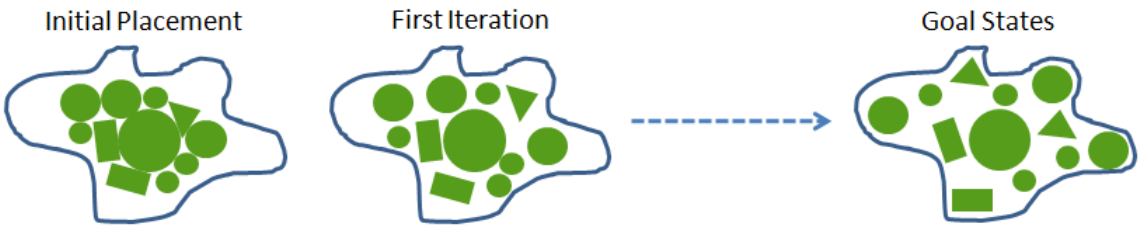


Figure 3.23: Bubble Packing Algorithm

Randomized Fractals Method

Randomized fractals methods are used to cover surfaces or volumes randomly with fractals. The main idea is to fill the fractals with the areas determined by the rule of [26] :

$$A_i = \frac{A}{\zeta(c,N)(i+N)^c} \tag{3.112}$$

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 defiend by

$$\zeta(c,N) = \sum_{i=0}^{\infty} \frac{1}{(i+N)^c} \tag{3.113}$$

It is possble to write,

$$\sum_{i=0}^{\infty} A_i = \sum_{i=0}^{\infty} \left(\frac{A}{\zeta(c,N)(i+N)^c} \right) \tag{3.114}$$

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 3.112, to cover the desired formation shape randomly.

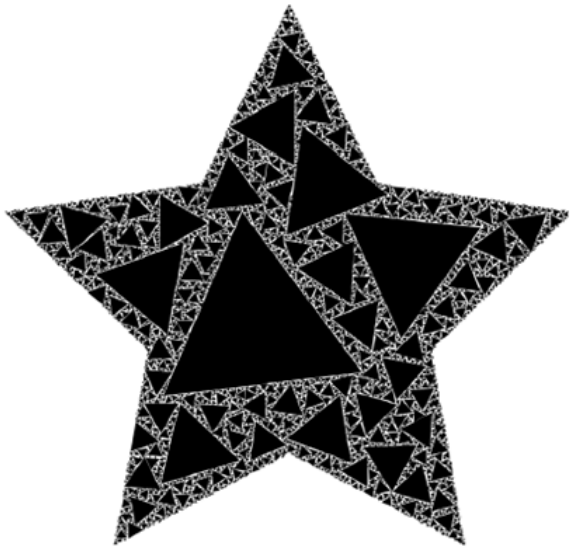


Figure 3.24: Randomized Fractals

Since the areas and the number of the agents, which will be represented with their coverage circles, are predetermined in our work, it is possible to adopt this 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 $C_{free} \neq Null$ **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

3.2.2.2 Decision Process on Goal States

In Section 3.2.2.1, the formation control problem is reduced down to a 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. The main approach to provide a solution for this problem is to make each agent to calculate the costs of its own to the goal states and then create a global consensus with the other agents to minimize the overall displacements of the swarm while achieving a desired formation shape.

In this work, the cost of reaching a goal state is defined with the minimum shortest path in the environment. A visibility graph based approach is used to calculate the shortest possible path to a goal state by taking into account the obstacles in the environment. The assignment process of the agents to the goal states which will be minimize the overall displacement, is handled with the help of Munkres (Hungarian) algorithm.

Free Configuration Space

Since the shortest path is defined in the free configuration space, each agent must calculate its free configuration space by taking the obstacles in the environment into the account.

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 environment 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.

$$C(R_i) = C_{free}(R_i, S) + C_{forb}(R_i, S) \tag{3.115}$$

If the forbidden space is calculated with Minkowski Sum method, free configuration space can be derived simply by extracting the forbidden space from the environment. 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,

$$S_1 \oplus S_2 := \left\{ p + q : p \subset S_1, q \subset S_2 \right\} \tag{3.116}$$

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

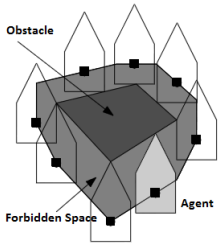


Figure 3.25: Forbidden Space Caused by an Obstacle

Forbidden space for agent i, $C_{forb}(R_i, S)$ is the sum of the forbidden spaces calculated for each obstacle in the environment.

Visibility Graphs and Dijkstra′s Algorihtm

It is obvious that the shortest path between two points in the environment must lie on the free configuration space, C_{free} for each agent to avoid collisions with the obstacle . A constraint for the shortest path is given as follows[30] :

> 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 $S^* := S \cup \left\{ p_{start}, p_{goal} \right\}$

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 vis-

ible (which are not crossing and interior region of an obstacel) connections between these nodes.

Consider set of obstacles in the environment is augmented with the Minkowski Sums. Let these set of augmented polygonal obstacles represented with $S_i \subset S$.

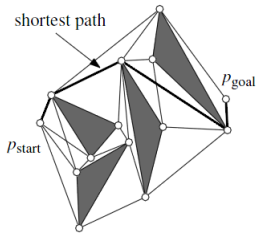


Figure 3.26: Shorthest Path from an initial state to a goal state

Algorithm to calculate the visibility graph of S^*,

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);

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. With the help of this $VISIBILITY\_GRAPH$ algorithm $SHORTEST\_PATH$

algorithm can be defined as follows.

Data: A set S of disjoint polygonal obstacles, and two points p_{start} and p_{goal} in the free space.

Result: The shortest collision-free path connecting p_{start} and p_{goal}

* Assign γ = VISIBILITY_GRAPH(S^*)

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

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

Algorithm 4: SHORTEST_PATH

Dijkstra′s algorithm computes the shortest path between two nodes in graph with k arcs, each having a non-negative weight. It is a tree search algorithm and used to calculate the shortest paths between nodes in a graphs, with the help of weighted edges between nodes. The 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. The algorithm for

the Dijkstra's is given as follows:

Data: γ_{vis} , *source_node*
Result: Shortest Distance to Any Node from *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 $v \subset \gamma_{vis}$;
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'S 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* .

In our work, the weights of the edges in γ_{vis} , are calculated with the Euler distance between nodes in the environment.

Collaborative Decision Process of Final Goal States

Possible routes and costs to the goal states are calculated with the help of Visibility Graphs and Dijkstra's algorithm for each agent. It is obvious that each agent will try

to choose a goal state with minimum cost according to its position and the orientation in the environment. But the cases in which two or more agents are willing to reach the same goal point must be handled by optimizing the overall utility of the swarm. To minimize to overall cost of whole swarm while achieving a formation shape, Hungarian algorithm which is a combinational optimization algorithm that solves this assignment problem is used. To implement this algorithm, a complete bipartite graph $G = (S,T,E)$ with $n \in S$ agents and $t \in T$ goal points is constructed.In this graph, each agent have a cost which is defined by the shortest path to the destination in the environment for different goal points.

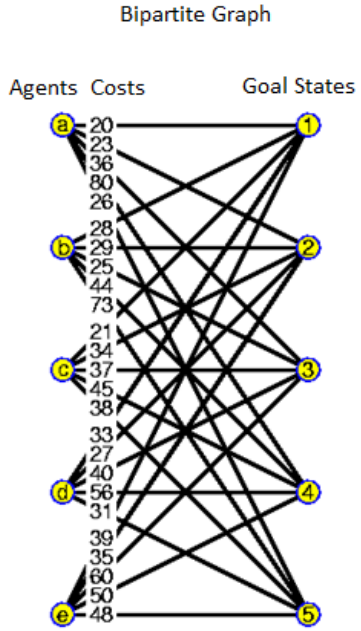


Figure 3.27: Sample Bipartite Graph Used in Assignment Problem

A cost matrix C is defined 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 matrix with nxn or mxm. 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

3.2.3 Mesh Quality Measurement

Since we have two different solutions to the shape partitioning problem, it is useful to define a method to compare the performances of these algorithms. One of the criterion of mesh diagrams is topological mesh irregularity [27-29]. This parameter

is defined by :

$$\varepsilon_t = \frac{1}{n}\sum_{i=0}^{n} |\gamma_i - D| \tag{3.117}$$

where

$$D = \begin{cases} 6 & \text{for triangles in Voronoi Diagram} \\ 12 & \text{for tetrahedras in Voronoi Diagram} \end{cases} \tag{3.118}$$

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. It is obvious that the topological mesh irregularity vanishes when all nodes have D neighbors, but it is almost not possible in practical application.

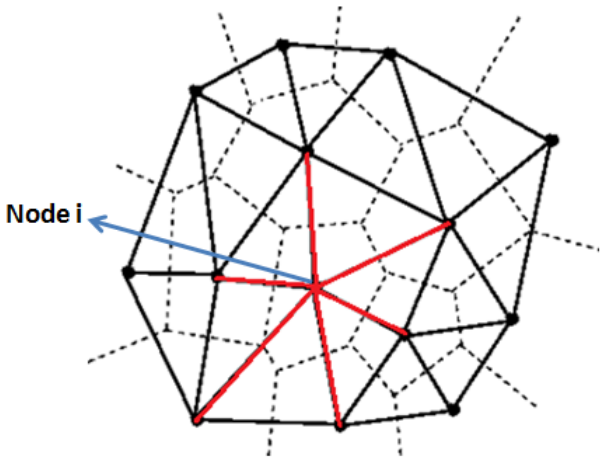


Figure 3.28: A Node with 6 Neighbors

Another metric to evaluate the quality or the performance of the mesh is the geometric irregularity defined as[27]:

$$\varepsilon_g = \frac{1}{m}\sum_{i=0}^{m} (A - \frac{r_i}{R_i}) \tag{3.119}$$

where

$$A = \begin{cases} 0.5 & \text{for triangles in Voronoi Diagram} \\ \sqrt{2/11} & \text{for tetrahedras in Voronoi Diagram} \end{cases} \tag{3.120}$$

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

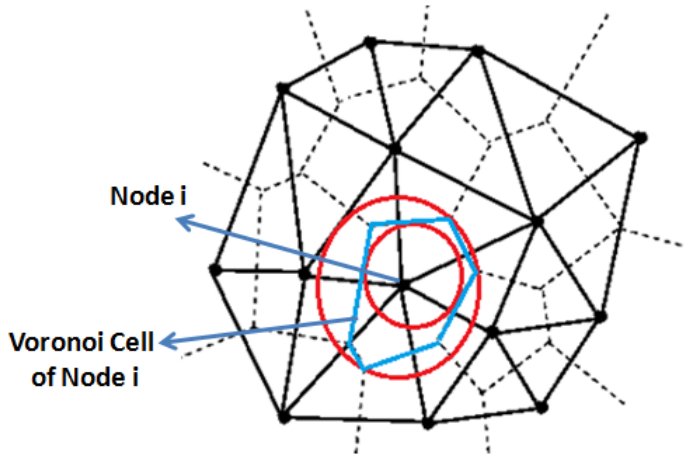


Figure 3.29: Inscribing and Circumscribing Circle of a Voronoi Cell Belonged to Node i

3.2.4 Control System Design for Individual Agents

Shape partitioning methods will provide the goal states for a desired formation shape and agents will be assigned to those goal states to minimize the overall energy consumption of the swarm. A control system which will guide the agents to reach these goal states must be designed for each agent individually. 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 iteration of formation control. On the other hand, the collision with the obstacles and the other agents at the environment must be prevented due the requirements presented in Section 1.3. The control system must react to these dynamic requests. For these purposes, a velocity controller with a large bandwith is designed at 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 3.30.

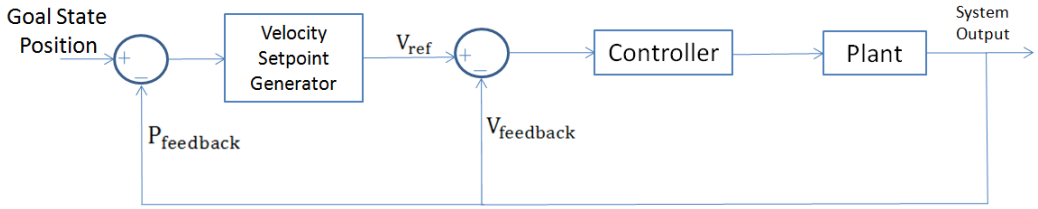


Figure 3.30: General Scheme of the Contol System

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 proportinal 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. This generator saturates the amplitude of the setpoint vector with 0.5 [m/sec] to prevent the agents' travelling in the environment so fast.

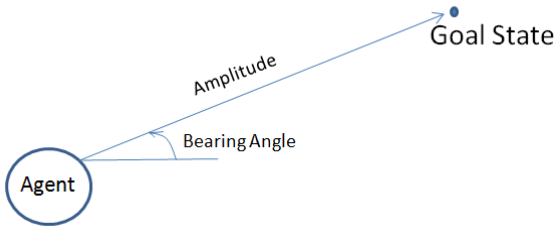


Figure 3.31: Velocity Setpoint Generation

The inner loop tis 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 an LQR controller. A simple mass, damper type second order linear system is used to provide a model for the controller. The translational friction force is assumed to be linear with the velocity of each agent. Different mass and friction coefficients for heterogenous 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,

$$\begin{bmatrix} \dot{v} \\ \dot{e} \end{bmatrix} = \begin{bmatrix} -b/m & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v \\ e \end{bmatrix} + \begin{bmatrix} 1/m \\ 0 \end{bmatrix} F_{net} \quad and \quad y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} v \\ e \end{bmatrix} \tag{3.121}$$

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

$$J = \int_{t_o}^{t_1} (x^T Q x + u^T R u) dt \tag{3.122}$$

is calculated with help of $'lqr'$ function of MATLAB for the given system in Equation 3.121 with the gain matrices of

$$Q = \begin{bmatrix} q_1 & 0 \\ 0 & q_2 \end{bmatrix} \quad and \quad R = r_1 \tag{3.123}$$

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

$$q_1 = \frac{1}{t_1 (x_{1max})^2}; \; q_2 = \frac{1}{t_2 (x_{2max})^2}; \; and \; r_1 = \frac{1}{(u_{1max})^2}; \tag{3.124}$$

Here t_i is the desired settling time for x_i which is determined as 1.5 seconds for the velocity and 0.001 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 **??**

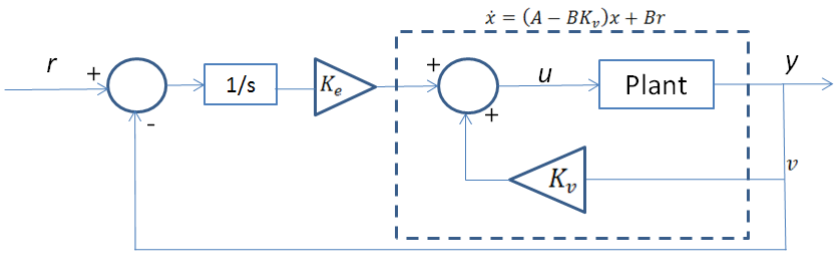


Figure 3.32: Inner Loop Structure

The error 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 velocity control input of the system.

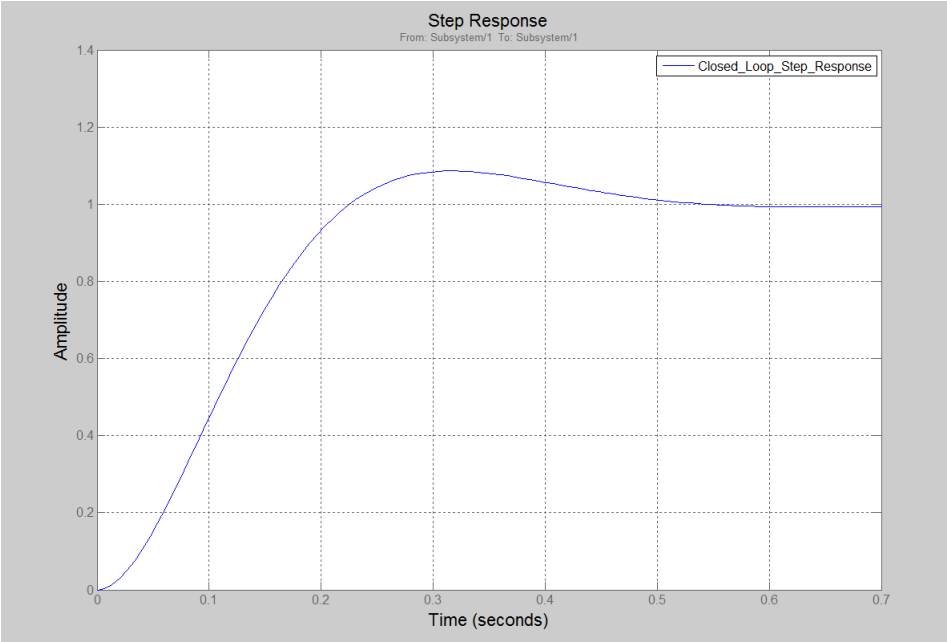


Figure 3.33: Step Response of the Closed Loop

According to the Figure **??**, the settling time for the inner loop is around 0.5 seconds for a step input. This response characteristics is important since the system cannot react to the setpoints changing faster than 0.5 seconds, so the formation control loops in which the new goal state assignments are handled must be executed with 2Hz frequency.

CHAPTER 4

NEXT CHAPTER

This is another chapter. All these chapters can go into seperate .tex files and you can include them with `\input{chapter1.tex}`. For only one citation [3], for multiple citations [1, 2, 4].

REFERENCES

[1] Darren C. Atkinson, Daniel C. Weeks, and John Noll. The design of evolutionary process modeling languages. In *APSEC*, pages 73–82. IEEE Computer Society, 2004.

[2] Thomas Erl. *Service-Oriented Architecture: Concepts, Technology, and Design*. Printice Hall, 2005.

[3] IAAS. Open source bpel4chor tools. https://github.com/IAAS, last visited on November 2013.

[4] Barbara Kitchenham and Stuart Charters. Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report, 2007.

APPENDIX A

APPENDIX NAME

Appendix content goes here.

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Surname, Name
Nationality: Turkish (TC)
Date and Place of Birth: dd.mm.yyyy, City
Marital Status: Single
Phone: 0 312 0000000
Fax: 0 312 0000000

EDUCATION

Degree	**Institution**	**Year of Graduation**
M.S.	M.S. Institute	M.S. Year
B.S.	B.S. Institute	B.S. Year
High School	High School Name	High School Graduating Year

PROFESSIONAL EXPERIENCE

Year	**Place**	**Enrollment**
Duration 1	Institute/Company 1	Role/Position/Experience 1
Duration 2	Institute/Company 2	Role/Position/Experience 2

PUBLICATIONS

International Conference Publications

Your publications goes here. Do not try to use Bibtex, since Bibtex builds a single bibliography database for the document.