A GRAPH THEORETIC APPROACH TO MULTI-ROBOT FORMATION CONTROL

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Electrical Engineering

by

Oscar Ruiz

June 2017

COMMITTEE MEMBERSHIP

TITLE: A Graph Theoretic Approach to Multi-Robot Formation Control

AUTHOR: Oscar Ruiz

DATE SUBMITTED: June 2017

COMMITTEE CHAIR: Xiao-Hua Yu, Ph.D.

Professor of Electrical Engineering

COMMITTEE MEMBER: Foaad Khosmood, Ph.D.

Professor of Computer Engineering

COMMITTEE MEMBER: Hisham Assal, Ph.D.

Lecturer of Computer Engineering

ABSTRACT

A Graph Theoretic Approach to Multi-Robot Formation Control

Oscar Ruiz

There are a variety of methods proposed for establishing a distributed controller for a multi-robot system. This thesis focuses on a graph theoretic approach to multi-robot formation control by exploiting the weighted consensus equation. We compare an absolute position based algorithm, derived from previous works, to an extension to the algorithm which removes dependence on an absolute frame of reference. The system is represented as a Graph and it is shown how the Graph Laplacian Matrix can describe control relevant information about the multi-robot network. The edges of the graph are assigned a virtual potential energy function and the total energy of the system is minimized by formulating our control strategy as a gradient descent algorithm. We also use the graph structure to specify robot formation and demonstrate the relationship between the Graph Laplacian Matrix and specific robot formations.

TABLE OF CONTENTS

Page

[LIST OF FIGURES v](#_Toc484452759)

[LIST OF TABLES vi](#_Toc484452760)

[Chapter 1: Introduction 1](#_Toc484452761)

[Chapter 2: Literature Review 5](#_Toc484452762)

[Chapter 3: Theoretical Background 9](#_Toc484452763)

[3.1 Graph Theory 9](#_Toc484452764)

[3.2 Edge Tension Energy Minimization 12](#_Toc484452765)

[3.3 Formation Matrices for Regular Polygons 18](#_Toc484452766)

[3.4 Avoiding False Formations 19](#_Toc484452767)

[3.5 Obstacle Avoidance 22](#_Toc484452768)

[3.6 Removing Dependence on Absolute Position 24](#_Toc484452769)

[Chapter 4: Simulation and Results 31](#_Toc484452770)

[4.1 Simulation controller 31](#_Toc484452771)

[4.2 Test Cases 33](#_Toc484452772)

[4.3 Simulation Results 34](#_Toc484452773)

[4.4 Comparison to Previous Implementations 41](#_Toc484452774)

[Chapter 5: Conclusions 46](#_Toc484452775)

[REFERENCES 47](#_Toc484452776)

[APPENDIX 49](#_Toc484452777)

# LIST OF FIGURES

Figure Page

Figure 1. Graph with Inter-Agent Tension Energy 13

Figure 2. Tension Energy Function 15

Figure 3. Formation Graph 18

Figure 4. False Formation Example 20

Figure 5. Obstacle Avoidance 23

Figure 6. Agent Network with Stationary Frame 25

Figure 7. Robot Network in Robot Frame 28

Figure 8. Formation to Regular Polygon with Eight Robots 35

Figure 9. Formation to Regular Polygon with 16 Robots 36

Figure 10. Formation to Customized Formations 36

Figure 11. Convergence Paths 37

Figure 12. Total Energy during Convergence 38

Figure 13. Total Energy during False Formation Avoidance 39

Figure 14. Maneuvering Around Static Obstacles 40

Figure 15. Total Energy during Static Obstacle Avoidance 40

Figure 16. Maneuvering Around Dynamic Obstacles 41

Figure 17. Total Energy during Dynamic Obstacle Avoidance 41

Figure 18. Mean Square Error for Inter-Robot Distances with K = 500 44

Figure 19. Error in Distances to Center of Mass 45

# LIST OF TABLES

Table Page

Table 1 - Pseudocode for False Formation Avoidance 21

Table 2 - Switching Conditions 31

Table 3 - Absolute Position and Reference Algorithm Comparison 44

# Chapter 1: Introduction

Swarm robotics is a method of coordination for multi robot systems, which are composed of large numbers of mostly simple robots [5]. The desired outcome is a collective behavior emerging from the interaction between the robots and their environment [12]. The robots are also autonomous, not controlled centrally and capable of local communication [13]. Swarm robotics has become a major research area since the 1980’s and new solution approaches are being developed and validated [13].  This thesis will focus on inter-robot motion coordination and some of the algorithms used for this task.

There are several active areas of research in the field of swarm robotics [5]. Swarm robotics and swarm intelligence is inspired by the decentralized organizational patterns found in the study of biology [10]. There are current attempts to adapt these natural processes to robot control and algorithm design. There is also on-going research in robotic swarm mobility, environment manipulation and reconfigurable robotics [10]. There are several key advantages to implementing a multi robot swarm as opposed to a single robotic platform. For situations where tasks can be decomposed into smaller subtasks, swarms can accomplish a given task more rapidly than a single robot by dividing the work into smaller parts and carrying out the subtasks in parallel [14]. Also, a robotic swarm can be designed in such a way that no single point of failure exists within the system. In other words, if an individual within the swarm becomes inoperable, the swarm itself should continue carrying out the tasks without interference. Related to this is the idea of scalability and locality. If more robots are added to the swarm, the individual’s already participating in the swarm should not be affected by the addition of new members. Also, individual robots within the swarm should be in communication with robots in their local vicinity and not with individuals that are far away and out of sight.

This is an important problem to solve because there are situations where it is advantageous to have multiple agents collaborating to accomplish a task as opposed to a single system. For example, the exploration of new environments could be done more rapidly with many smaller, simpler and cheaper robots as opposed one large expensive one. The multi-robot system could explore a larger area and be less vulnerable to mission failure if one of the members of the system becomes inoperable. Another valuable application of swarm robotics is in search and rescue. Small robots, with swarming capabilities can search destroyed buildings for survivors by reaching places that are out of reach by human rescuers. There are also applications in formation control for satellite clusters. The Terrestrial Planet Finder is a proposed deep space interferometer composed of multiple telescopes which stay in formation. Communication between the platforms is essential and the control of the distances between the telescope is important for correct operation of the telescope.

This thesis studies robotic swarm motion coordination and how robots within a swarm calculate and adjust their positions with respect to each other using a decentralized control algorithm. The goal of this project is to replicate the control strategy introduced in resources [1] and [2] and to develop a fixed frame independent control algorithm, which allows for autonomous control of a multi-agent system. This is needed in the control of robotic swarms where we have a distributed network of simple robots, which have limited communication capabilities and should be independent of a central command location and reference point.

The rendezvous problem is modeled and solved using the algorithm in [1] and modifications are introduced to allow for greater control over a particle swarm including predefined formation achievement and collision avoidance. This is accomplished via the consensus equation as applied to a multi-agent system. Weights are applied to the consensus equation to prevent inter-agent collisions. Finally, the algorithm is expanded to remove the dependence of information about absolute position in the system. The modified algorithm instead relies on inter agent distance and orientation. We compared the performance of the Absolute Position dependent algorithm and the Robot Frame based algorithm in terms of their ability to accomplish certain tasks.

This thesis is organized as follows. In Chapter 1 the problem is established and reasons for its importance are given. Chapter 2 covers an overview of the field of swarm robotics. It introduces the main sources for the algorithm derived in this thesis as well as other strategies attempted by different research groups. The derivation of the algorithm begins in Chapter 3 by replicating the system equation from the Graph Laplacian based on [1], [2] and [6]. The derived system equation includes capabilities for system convergence to a location and achieving formations based on the weighted consensus equation. Once we establish a way to maintain a constant inter-agent system, we introduce an extension to the strategy that includes obstacle avoidance and local minimum avoidance. In the Chapter 4, we introduce test cases and compared the performance of the original control system developed in [1] and [2] to our Robot Frame based control strategy. We define a set of objectives the systems should accomplish including: rendezvous and formation, maneuvering through a set of static obstacles, and maneuvering through a set of dynamic obstacles. We also compare the performance of the Absolute Position Strategy to the results presented in our main reference paper [1]. Finally, in Chapter 5 we present our conclusions and described our intentions with the continuation of the project, including possible improvements to the algorithm and the implementation of the algorithm on physical hardware.

# Chapter 2: Literature Review

We found several Graph Theoretic Methods for establishing multi-agent control systems. Some of the other methods we found were based on state switching algorithms without reference to a graph structure such as [4], as well as a swarming method based on real time learning using an embedded neural network [5]. Since our method is based on Graph Theory we reference those papers throughout this thesis. We will look at those alternative method briefly in this literature review.

*Edge-weighted consensus-based formation control strategy with collision avoidance* was our main reference and source of inspiration for this thesis. The model described in this thesis is based on the decentralized control strategy from [1]. The paper describes the graph based structure of a multi-agent system in terms of edges and vertices as ours does. Our method diverges from the one found in [1] in the definition of the Graph Laplacian Matrix. They define their L matrix as the Gramian of the Incidence Matrix while we describe our L matrix as the differences between the Degree and Adjacency matrix. Both definitions are equivalent. Our definition of the L matrix has the advantage of giving as access to the Adjacency and Degree values directly which become useful for building prescribed formations in section 3.3

*Graph-Theoretic Connectivity Control of Mobile Robot Networks*[2] was also important to our understanding of Algebraic Graph Theory as it relates to robotic networks. This paper provides a definition for connectivity based in Graph Theory. The paper provides the derivations for the Weighted Graph Laplacian as described in this thesis and introduces a theoretical framework for insuring graph connectivity in mobile robot networks where such connections are not guaranteed due to line of sight or sensor range limitations. This is distinct from our method in that ours assumes connectivity always regardless of range or loss of line of sight.

*Decentralized Rendezvous of Nonholonomic Robots with Sensing and Connectivity Constraints [3]* introduces a similar approach to our method while leaving out a reference to graph networks. This paper [3] considers a group of wheeled robots with non-holonomic constraints and a method for rendezvous at a common specified point with prescribed orientation while also maintaining network connectivity and ensuring collision avoidance within robots. This method is different from the strategy in this thesis because they do not invoke Graph Theory and instead derive their algorithm from the dynamics of the system. However, the paper does consider the system equation as a gradient descent of a potential function as is the case in our thesis.

*Distributed Colony-Level Algorithm Switching for Robot Foraging [4]* was the first reference paper we studied to attempt to replicate previous results in the field of multi-robot systems. This paper [4] focuses on foraging as a multi-robot task and presents two distributed foraging algorithm each of which performs best for different food distributions and locations. A third algorithm with the ability switch between the two previous algorithms is also presented. The method is based on state switching of each individual robot based on the conditions around it. The first method is referred to as the Gradient Method where robots perform a random walk and switch to beacons at random moments in time. The beacons broadcast a number representing the gradient towards the nest. When the resource is found the bots know in which direction it is based on the gradient established. The second algorithm is based on a sweeping method where the bots form a line, hold in place based on a virtual force between them and circle around the nest to look for the resource. The virtual force described in this paper is like our energy function used in this thesis. The third algorithm switches between the two previous methods based on whether a given method is having success at finding the resource. We used the algorithm in [4] to do early modeling in python but eventually moved towards a graph based model.

*Embedded Neural Network for Swarm Learning of Physical Robots [5]* was distinct from the other literature we looked in its use of an embedded neural network as a control system. In this paper [5] real time learning of multiple physical autonomous robots situated in a real dynamic environment was performed. Each robot had an onboard micro-controller where a simple artificial neural network ANN was embedded. The system was designed with consideration of the power and computational resource limitations of the robots. The robot ANN’s were started simultaneously and ran until the desired inter-robot distances were achieved. This method depended on the perimeter of the arena to control for distances and was not a robust as some of the other strategies we looked at. However, the prospect of implementing real-time learning capabilities along with reactive distributed graph based systems is something we’d like to experiment with in future works.

*Hybrid Control for Connectivity Preserving Flocking [6]* is like the research in [2]. It also has many of the same authors. In this paper [6] the same problem of motion and network topology control in a group of mobile agents is explored but instead of assuming that the graph is connected, it is enforced through distributed topology control which decides on both direction and creation/deletion of links. Our algorithm assumes connectivity always and has pre-supposed link deletions based on the formation we would like to achieve prior to starting the simulation.

*A Control Allocation Approach for Energetic Swarm Control [7]* uses the analogies of force and energy for inter-robot interactions as is done in this thesis. This paper [7] also invokes a low-level trajectory controller based on dynamic feedback linearization. This approach was not used in this thesis but provides an example of what others have done for the problem of creating a high level multi agent control algorithm and interfacing it with a low-level controller for individual robots.

# Chapter 3: Theoretical Background

## 3.1 Graph Theory

The following section presents a model of our multi-agent system in the context of Graph Theory and its mathematical structures based on [2] and [1]. We will be using the model described in the preliminary section of [1].

Graphs are composed of vertices connected by edges. Graphs can be directed or undirected. Undirected graphs do not make a distinction between two edges connected to the same two vertices while directed graphs do. We can represent our multi-agent system as a graph with the set of vertices as agents and the set of edges as the communication links between agents.

|  |  |  |
| --- | --- | --- |
|  |  | (1) |
|  |  |  |
|  | *is the Vertex set* | (2) |
|  |  |  |
|  | *is the Edge set* | (3) |

For undirected graphs, the edge  connecting vertices is indistinguishable from edge . The graph representing our multi-robot system will be undirected because we do not care about the direction of communication between two robots. We are only concerned with the existence of connections between robots. Furthermore, we define  to be the set of vertices that are connected to vertex  . We also call this set the neighbors of vertex  . We can now define the Adjacency matrix of the graph as A. The Adjacency Matrix is a symmetrical matrix with diagonals equal to zero.

|  |  |  |
| --- | --- | --- |
|  |  | (4) |
|  |  | (5) |
|  |  | (6) |

The Adjacency matrix defines which vertices are connected and which vertices are disconnected through their corresponding graph edges. The elementin A is equal to 1 when the edge exists and is equal to 0 when the connection edge does not exist. In other words:

|  |  |  |
| --- | --- | --- |
|  | if | (7) |

We can also define the Degree matrix of the graph. The Degree matrix is a diagonal matrix with the element  being the number of connections (the degree) to the vertex.

|  |  |  |
| --- | --- | --- |
|  |  | (8) |

The diagonal elements of the degree matrix are also equal to the sum of the row of the corresponding Adjacency matrix. In other words:

|  |  |  |
| --- | --- | --- |
|  |  | (9) |

Using the Degree Matrix and Adjacency Matrix we can define the Graph Laplacian as:

|  |  |  |
| --- | --- | --- |
|  |  | (10) |

The Laplacian of the graph defines the uniqueness of the graph and can be used to define the state equation of our multi-robot system [1], [2], [6].

We can write the state of the total multi-robot system as:

|  |  |  |
| --- | --- | --- |
|  |  | (11) |

In our system,is the displacement with respect to a fixed frame of robot  in a one-dimensional environment. Also, n is the total number of robots in the system. For simplicity, we will continue our analysis in one dimension but our derivation can be generalized to multiple dimensions. We can begin prescribing certain behaviors to our system that would allow us to more easily control their formation. We can begin with the elementary approach of driving two robots to a single point.

|  |  |  |
| --- | --- | --- |
|  |  | (12) |
|  |  | (13) |

The above equations of motion will cause the robots to drive towards each other until the distance between them is zero. If we extend this to  robots the equation of motion for the first robot is:

|  |  |  |
| --- | --- | --- |
|  |  | (14) |

The above equation assumes that the graph is fully connected. We can restrict the connections for each robot to a certain set which we will call its neighbors . A robot’s set of neighbors can be defined by proximity or through some other imposed means such as for the purposes of establishing a formation. When robot neighbors are defined, the equations of motion can be written as:

|  |  |  |
| --- | --- | --- |
|  |  | (15) |

We can also write the consensus equation in terms of the of the number of connections to the given robot and the elements of the adjacency matrix that are non-zero.

|  |  |  |
| --- | --- | --- |
|  |  | (16) |

Therefore, the equation of motion for the entire system can be written in terms of the Adjacency and Degree matrices.

|  |  |  |
| --- | --- | --- |
|  |  | (17) |

And in terms of the Graph Laplacian the System equation is:

|  |  |  |
| --- | --- | --- |
|  |  | (18) |

This gives a compact form for the behavior of the multi-agent system including robot positions and communication links between robots that guarantees the system will converge. This guarantees the system will converge but it is a naive control strategy because it will drive all agents to single point. In the next section, we will adjust the equation to prevent inter-robot collisions by introducing weighting functions.

## 3.2 Edge Tension Energy Minimization

The previous equation of motion solves the problem of achieving robot rendezvous at a single point. We can introduce weight functions that repel the agents as they get close to each other to prevent them from colliding with each other as well as maintain a constant inter-agent distance. This approach can be found in [1] and [2].

|  |  |  |
| --- | --- | --- |
|  |  | (19) |

The weights are functions of the Euclidean norm (distance) of the positions between two adjacent agents. The distance is defined as:

|  |  |  |
| --- | --- | --- |
|  |  | (20) |

We would like to derive our weight functions to prevent agent collisions and maintain constant inter-robot distances. To begin our derivation, we can define a virtual tension energy between two agents as  which is a function of the distances between two agents.

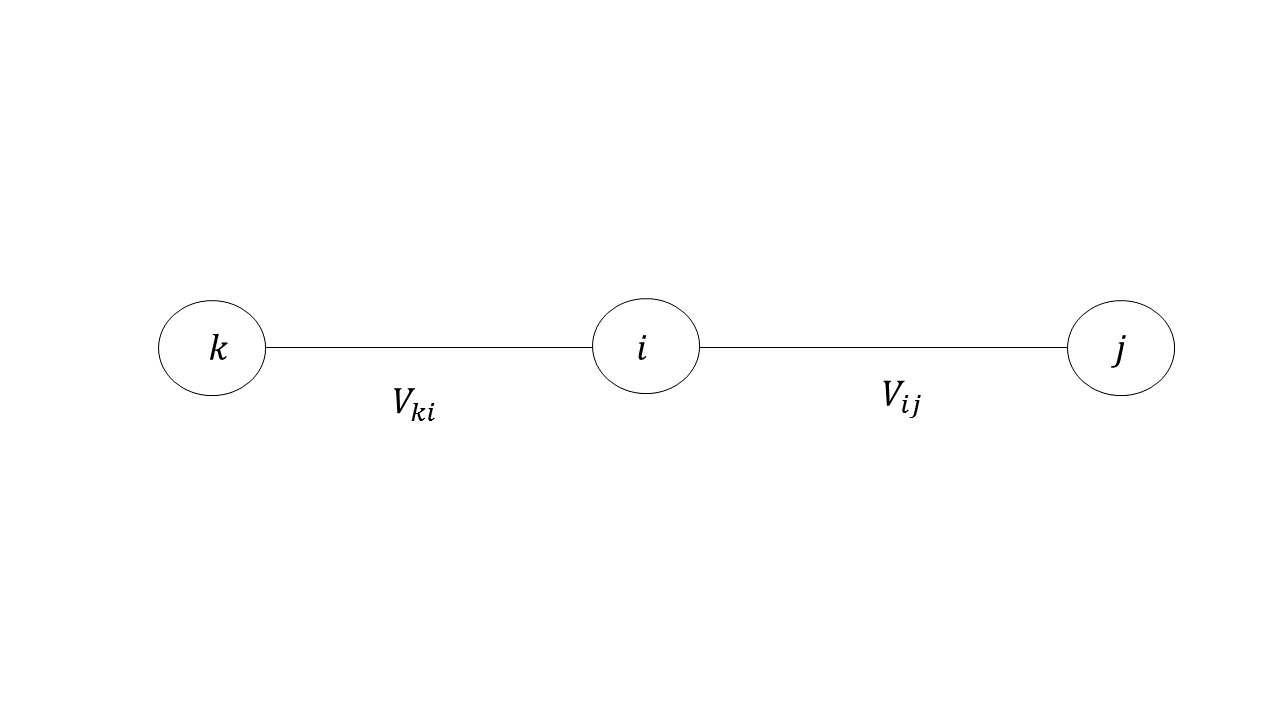


Figure 1. Graph with Inter-Agent Tension Energy

Figure 1. Shows an example of a multi-robot system consistent of agents i-j-k with inter agent tension energies. The total energy of the system can be written as:

|  |  |  |
| --- | --- | --- |
|  |  | (21) |

We divided the sum in half because the System’s Graph is undirected and therefore:

|  |  |  |
| --- | --- | --- |
|  |  | (22) |

We rewrote our system equation as a gradient descent algorithm where an agent position is found to minimize :

|  |  |  |
| --- | --- | --- |
|  |  | (23) |

The desired outcome is achieved because the derivative of the total energy with respect to time is always negative and therefore the energy is always decreasing.

|  |  |  |
| --- | --- | --- |
|  |  | (24) |

Now the challenge is to find weight functions and objective functions for the energies that would make our gradient descent equation and our consensus equation equivalent. A straightforward way is to choose an energy function which is a function of the inter-robot distances and has single distinct global minimum which the gradient descent equation can use to compel robot positions towards. For these reasons, we chose a quadratic function of the form:

|  |  |  |
| --- | --- | --- |
|  |  | (25) |

In equation (25), is the immediate relative inter-robot distance between robot  and . Also, is the desired inter-robot distance for all robots under the influence of this energy function.

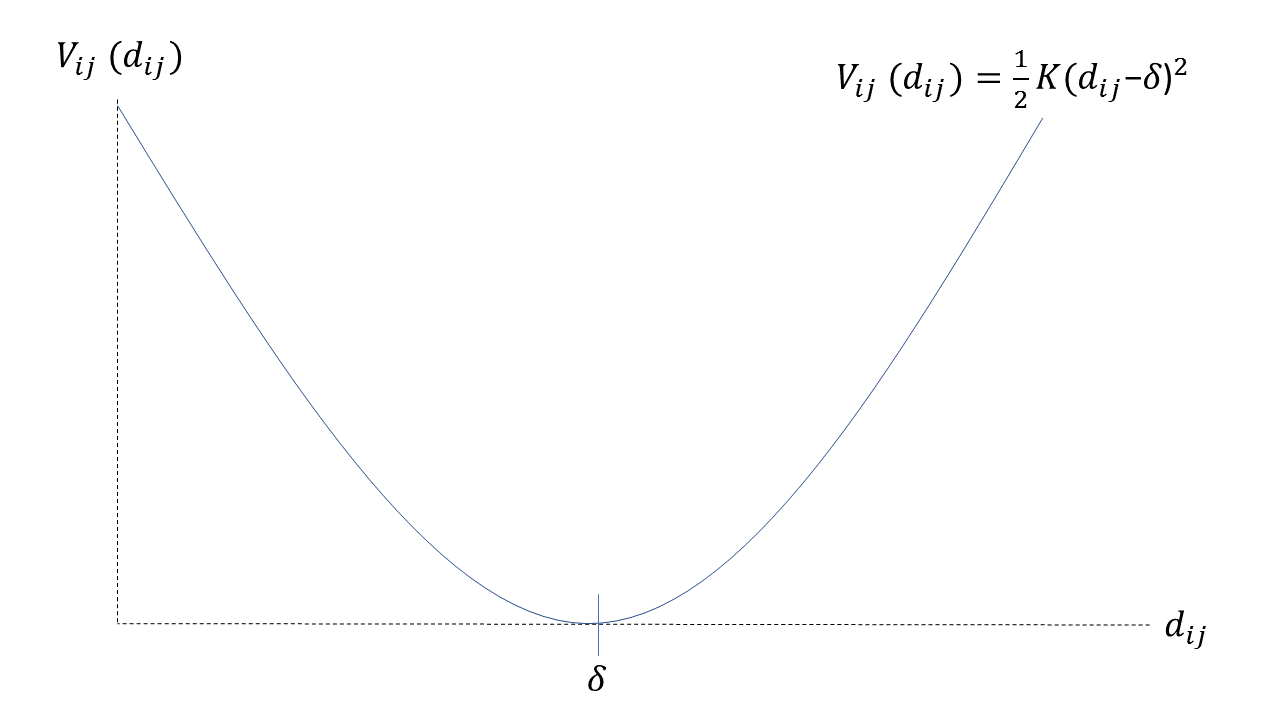


Figure 2. Tension Energy Function

Figure 2 shows the form of the energy function. The tension energy function is quadratic and the global minimum of the function can be found at . We can use a gradient descent algorithm to find the function’s minimum. If we take the derivate of the potential with respect to we get:

|  |  |  |
| --- | --- | --- |
|  |  | (26) |

We know that because the partial derivate of the Euclidean norm can be shown to be:

|  |  |  |
| --- | --- | --- |
|  |  | (27) |

Therefore, equation (19) can be rewritten as:

|  |  |  |
| --- | --- | --- |
|  |  | (28) |

The weight functions are written as:

|  |  |  |
| --- | --- | --- |
|  |  | (29) |

By writing the control algorithm as a gradient descent, two agents moving away from each other will undergo an attractive virtual force from the positive gradient of . They will likewise be repelled by the negative gradient when . The agent will stop moving when .

With our new system equation, we can define new Weighted Adjacency and Degree Matrices. The Weighted Degree Matrix is a diagonal matrix with the element  being sum of the weights  where s is the degree of  () if the edge exists. By including the elements of the adjacency matrix we ensure that links that do not exist are not included in the sums. We also do not include values where  because the link connecting an agent to itself is meaningless in our application. This definition for weighted degree matrix is found in [2].

|  |  |  |
| --- | --- | --- |
|  |  | (30) |

The Weighted Adjacency Matrix has the same structure as the Adjacency Matrix except the connected elements where are replaced with our weight functions .

|  |  |  |
| --- | --- | --- |
|  |  | (31) |

Our Weighted System equation can be defined in terms of the Weighted Graph Laplacian matrix which is the difference between the weighted Degree Matrix and Weighted Adjacency Matrix:

|  |  |  |
| --- | --- | --- |
|  |  | (32) |

This again gives a compact description for the behavior of the system:

|  |  |  |
| --- | --- | --- |
|  |  | (33) |

Equation (33) can be compared to equation (18) which is the unweighted dynamic equation for the multi-robot system. The above equation describes the behavior of the multi-robot system where each robot has an equation of motion in the form of equation (19). And the total system is guided by the gradient descent algorithm, equation (23). The total energy of the system is minimized when the inter-robot distances are equal to the prescribed distance . This allows us to space the robots apart and maintain that distance under perturbations.

However, is obvious that not all robots can be the same distances from all other robots simultaneously, which means the total energy of the system will never actually reach zero. To control for a formation, we will need to restrict the allowed edges between vertices. This can be done by creating custom Weighted Graph Laplacian Matrices that define our target formations. In the following section, we will describe how we can choose values for the Weighted Adjacency and Weighted Degree matrix to create certain unique shapes with the multi-robot system.

## 3.3 Formation Matrices for Regular Polygons

We looked at several graph structures and their relationships to robot formation in the context of the control scheme we have developed. Based on the definitions for graphs discussed in section 3.1 we can define Graphs that would give exact geometric formations in two-dimensional space. We will proceed to build regular polygons with different number of vertices. To avoid fix formatting the values of the graph and allow for a more dynamic behavior of the system, we will only be assigning connections to the agents that should be immediately adjacent to each other and the next diagonal over as shown in the figure 3.3:

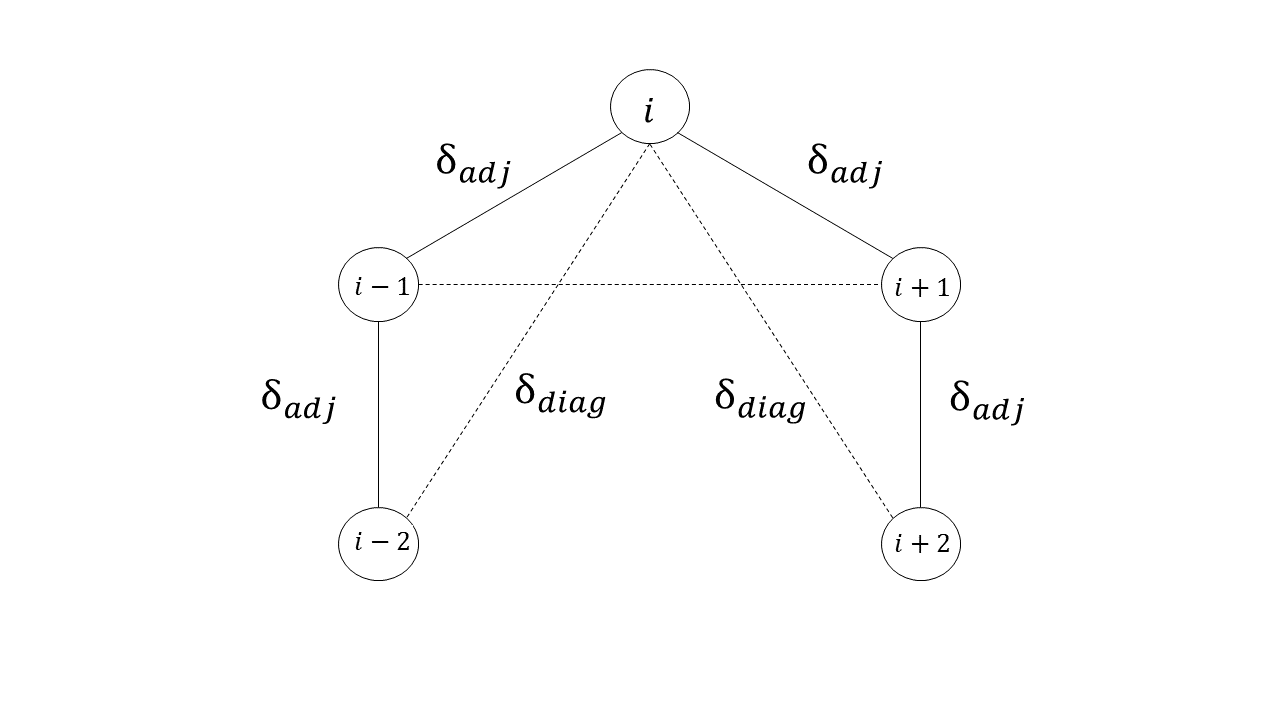


Figure 3. Formation Graph

Figure 3 shows the graph for the pre-established connections in our multi-agent system. All other agents will be disconnected initially to allow the system to contract and expand as needed. For regular polygonal shapes, the edges immediately adjacent to each vertex can have length . In figure 3, these vertices have index and relative to a given vertex. The non-adjacent connected vertices are known as the diagonal vertices. The diagonal vertices are connected edges with lengths and have indices  and  relative to vertex  as shown in figure 3. The required distance for the diagonal edges can be found by the formula [19]:

|  |  |  |
| --- | --- | --- |
|  |  | (36) |

In equation (36), n is the number of vertices in the graph. When defining our Weighted Adjacency Matrix  the input for its weights (equation 29) will beand . We can now define an algorithm for defining our weight functions (equation 29) for any value of n. If the weights in our system equation have the general form:

|  |  |  |
| --- | --- | --- |
|  |  | (37) |

The value of  will be:

if the agents are immediately adjacent.

if the agents are diagonally adjacent.

This determines the weight functions for robots that are adjacent to each other in the regular polygon and for the first cross-diagonal robots. Fully connected graphs describing a multi-robot system are defined as graphs where every robot  is connected to every other robot in the system. Graphs for robot systems composed of less than 6 robots will be fully connected. For larger systems, there will be gaps in the connection of the graph that will need to be accounted for when the system becomes tangled or falls into a false formation.

## 3.4 Avoiding False Formations

When defining the Weight Matrices for our robot formations, we established connections between adjacent robots with edges corresponding to sides and cross diagonals for regular polygons. This means that the graph is fully connected for up to n = 5 robots. For larger systems, some of the vertices in the graph are disconnected from robots outside of their immediate vicinities. This means that robots that do not follow under the category of adjacent or diagonally adjacent are not connected. The problem arises for larger collections of robots, where all the inter-robot target distances for connected robots could be potentially satisfied without achieving the desired polygon formation. This gives a false formation or a local minimum in the total energy and needs to be accounted for.

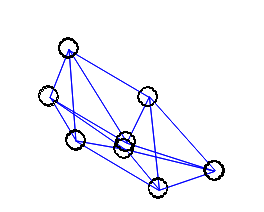


Figure 4. False Formation Example

Figure 4 shows an instance where two or more robots ended up in the incorrect in positions relative to the rest of the system. To compensate for this, we establish temporary connections between robots that are not already connected by the Weighted Graph Laplacian. The temporary connections can be activated whenever unconnected robots move within in a certain activation distance  of each other. The temporary connection creates an energy tension of the same form as the Graph’s tension parabolic function but drops to zero beyond the vertex of the parabola, as in the second equation in (38). This creates a repulsive force below that activation distance but does not attract beyond that activation distance.

|  |  |  |
| --- | --- | --- |
|  | if and  if | (38) |

We do not know what the activation distance should be beforehand for a given multi-robot system. Each robot requires a separate target distance to create regular polygon formations for larger number of robots. We developed an algorithm that resets to allow the formation to avoid a local minimum. We can set where  and  are the numerator and denominator of the activation distance respectively and have no physical meaning. Furthermore, we initialized the activation distance to some arbitrarily large number. In our simulation, we set the initial value ofwhere n is the number of robots and is the side length of the target polygonal formation from equation (37) and its input .

Throughout the formation processes we want the energy to decrease. If for any reason the energy begins to increase or stops decreasing, it means that is too large and is keeping the robots from getting to their target inter-robot distances. While this is the case, b begins to increase, reducing the size of , and a counter c begins to count. In the best-case scenario, is reduced to the point where it is no longer active, meaning  in equation (38). In this case, the total energy resumes its decline and reaches a global minimum. If the change in total energy is positive by the time the counter  reaches its maximum value, , it means that the system has fallen into a false formation. In this case,  is set to 1,  is reset to 0 and  is increased. This increases the size of with the intention of restarting the formation the process. This algorithm will repeat until  reaches its maximum value, , and the whole cycle repeats. The algorithm can be summarized with the following pseudo-code:

Table 1. Pseudocode for False Formation Avoidance

|  |
| --- |
| 1 if  2  =  + β  3  =  + 1  4 if c > :  5  = 1  6  = 0  7  =  + 1  8 if  >:  9  = 1 |

In summary, every time nonconnected agents get closer to each other than the activation distance, and ,  is increased so that  decreased. The activation distance has a maximum allowed value () so that it does not increase indefinitely and cause the system to diverge. It should be noted that every robot needs access to the value of the total energy of the system. Therefore, it is necessary for the graph of the system to be connected so that the value of the energy can be broadcasted to everyone. The results of the false formation avoidance technique are described in Chapter 4.

## 3.5 Obstacle Avoidance

Now that we have a method for achieving a set formation based on the prescribed graph and edge weights for inter-robot distance, we would like to expand our controller to allow the system to avoid obstacles as it moves through a space. We can do this by deriving weight functions that are functions of the distances between robots and obstacles.

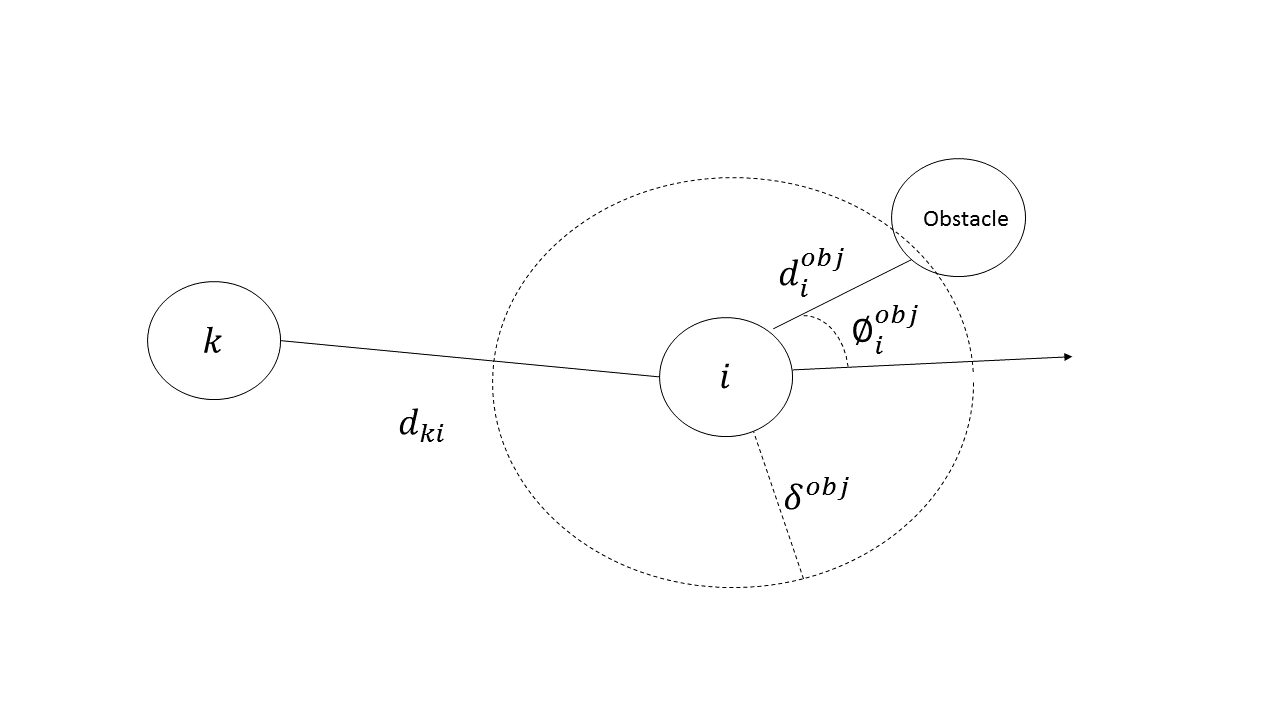


Figure 5. Obstacle Avoidance

Figure 5 above shows a model of an agent interacting with an obstacle and its respective inputs*.* In the same way that we derived the weights for inter-robot coordination, we can define a tension energy between the obstacle and weight that has the form:

|  |  |  |
| --- | --- | --- |
|  |  | (39) |

In figure 5, is the distance from the robot to the object and is the range of activation for . The weights are activated when the obstacle is in the range of :

|  |  |  |
| --- | --- | --- |
|  | If and  If | (40) |

We chose this scheme because the value will have a repelling effect on the robot’s equation of motion with respect to the obstacle when the obstacle is within the range of but it will be zero or non-attractive when it is beyond the range of the . It plays a similar role to in equation (38) and section 3.4, except it is static.

The weight terms are summed over the number of objects in range of robot .Our modified dynamic equation now includes the weighted consensus equation (19) and obstacle avoidance:

|  |  |  |
| --- | --- | --- |
|  |  | (41) |

In the equation above, is the position of the detected obstacle and is the set of obstacle vertices that are connected to the vertex  (i.e. ). For the purposes of experimentation in Chapter 4, we will add a global offset term  that moves the system through a set of obstacles:

|  |  |  |
| --- | --- | --- |
|  |  | (42) |

This will not guarantee the formation will be held once the system moves through the obstacles but is simply to compel the system to move in a desired general direction. We have now derived a total system equation of motion that can maintain formation, avoid obstacles and move in a desired direction. The limitation of this strategy is that we need the exact location of all the agents, the obstacles and the target location relative to a fixed reference frame. In the next section, we attempt to reformulate our system equation to remove any dependence on absolute position and depend, instead, only on distance and orientation with respect to the robots in the system.

## 3.6 Removing Dependence on Absolute Position

Let us assume that our multi-agent system is composed of n robots and let us also consider robots , and  for now. We will consider robot  to be the primary robot under analysis.  This robot has a direction heading defined by .

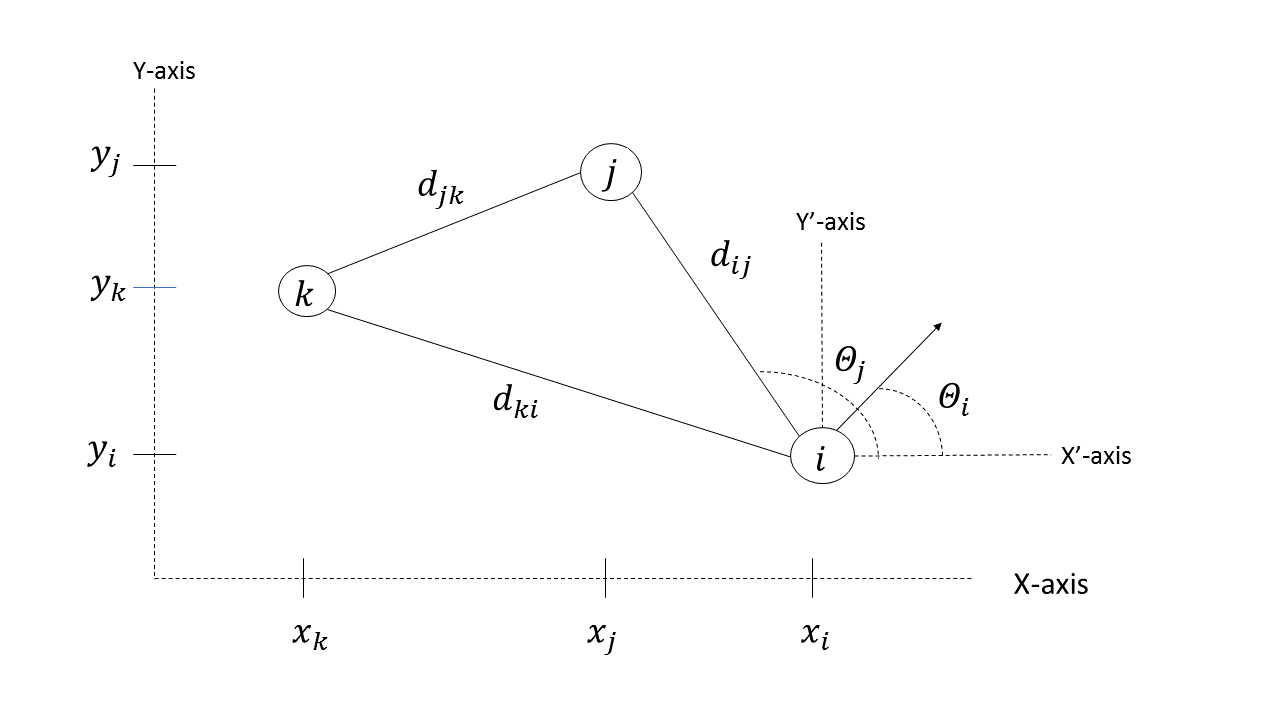
**

Figure 6. Agent Network with Stationary Frame

Figure 6 shows robots , and  with their respective coordinate positions,  and with respect to fixed frame . A rotation-less frame is defined as and is fixed to the body of robot , which is the robot we will perform analysis on. Robot  has a heading vector which is angle with respect to the x’-axis. We can see from the figure that the difference between horizontal coordinates and  can be written in terms of the Euclidean distance between the robot under consideration and the angle made between the x ‘-axis and the direction to that robot:

|  |  |  |
| --- | --- | --- |
|  |  | (43) |

We can replace the terms  in our weighted consensus equation with equation (43).

|  |  |  |
| --- | --- | --- |
|  |  | (44) |

In figure 6, the angle  is the angle made between the x’ axis and the direction towards robot . Also is angle difference between  and  or, in other words, is the angle robot  needs to turn to face robot .

|  |  |  |
| --- | --- | --- |
|  |  | (45) |

If we substitute equation (45) into equation (44) and invoke the trigonometric identity for angle sums, we get:

|  |  |  |
| --- | --- | --- |
|  |  | (46) |
|  |  | (47) |

We can rearrange the equation so that we can isolate the terms involving the parameters dependent on a fixed reference frame:

|  |  |  |
| --- | --- | --- |
|  |  | (48) |

We can repeat this process for our vertical terms using the trigonometric identities for the  function:

|  |  |  |
| --- | --- | --- |
|  |  | (49) |

As with our previous model the heading a robot needs to move towards to satisfy the consensus equation will be induced by the outputs of the consensus equation  and . This heading is angle  and is defined to be with respect to the x axis of the rotation-less reference frame F’. If the robot does not need to move, then these outputs are zero and the desired heading is not defined.

|  |  |  |
| --- | --- | --- |
|  |  | (50) |

The required angle  is equal to the heading of the analyzed robot plus some angle the robot needs to turn to achieve the required angle.

|  |  |  |
| --- | --- | --- |
|  |  | (51) |

If we substitute equations (48), (49) and (51) into (50) we get:

|  |  |  |
| --- | --- | --- |
|  |  | (52) |

We can once again invoke trigonometric identities to change the LHS:

|  |  |
| --- | --- |
|  | (53) |

We would like to write our control equation independent of variables referencing the rotation-less frame F’ on robot  or the space fixed frame F. If we compare terms in equation (53) we can see that:

|  |  |  |
| --- | --- | --- |
|  |  | (54) |
|  |  | (55) |

The angle  is the angle robot  will have to turn to be on the correct heading  to satisfy the weighted consensus equation (19) from section 3.2. The angle  is defined by the equation:

|  |  |  |
| --- | --- | --- |
|  |  | (56) |

We can see that equation (56) is depending only on inter robot distances which can be taken from distance sensors, and the angle the neighboring robots make with robot ’s heading. With this information, we can continually update the required heading of each robot in a decentralized and frame independent manner.

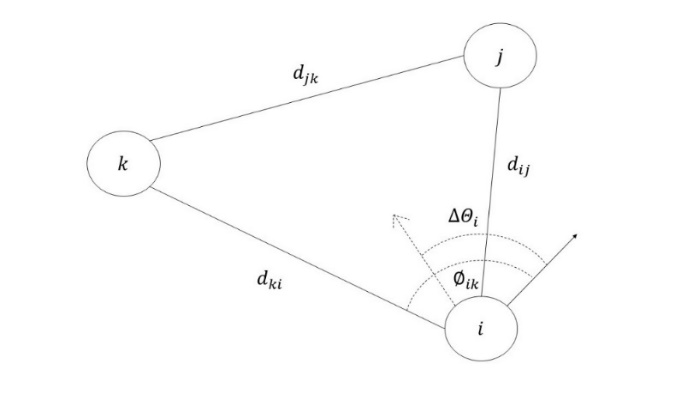


Figure 7. Robot Network in Robot Frame

Figure 7 shows the robot network with distance and angel inputs independent of a stationary frame. This method, however, does not give us the required speeds of each robot. We need a way to tell a robot to stop when it has reached its target location and not move when it is oriented in the wrong direction. One approach we can take is make the speed of the robots proportional to the tension energy between its neighbors. By doing so, as the energy becomes large, the robots will speed up and as the energy approaches zero they will slow down. We will also set a maximum value for the speed to prevent the system from diverging.

|  |  |  |
| --- | --- | --- |
|  |  | (57) |

This method was inspired from reading on the adaptive gradient descent algorithm where the iteration rate becomes a function of the gradient instead of a constant [8]. However, instead of using the gradient we are using the edge weight tension energy to change the iteration rate. The information collected from  and  can be used to update the positions of all robots in our simulation. We also expanded our control strategy by adding a term for obstacle avoidance. To do this, we can add new rotation commands to update the angle for our robots which respond to the weights and distances to the obstacles. The rotation commands are calculated in the same way as in equation (56):

|  |  |  |
| --- | --- | --- |
|  |  | (58) |

In the above equation,  is the distance from robot to an obstacle. Also, is the angle difference between the direction to the obstacle and the robot’s heading as defined in figure 5. Finally, we can include a term for moving towards a target location:

|  |  |  |
| --- | --- | --- |
|  |  | (59) |

This also means we need to expand our term for desired speeds for the Robot Frame Algorithm. The calculation for desired speeds will include the relevant energies associated with robot-obstacle interactions and distance to the target location:

|  |  |  |
| --- | --- | --- |
|  |  | (60) |

The maximum value for speed was determined through trial and error in our experiments. We set our max speed to 40,000 units when the time step is 0.00001 and found good results which are shown in Chapter 5. The constants ,  and are scaling constants the were also determined experimentally and are switched on and off according to a switch case in Table 2. The total rotation angle can be written as:

|  |  |  |
| --- | --- | --- |
|  |  | (61) |

In equation (61), is the same term as defined in equation (56). We can add the rotation angles because the total required change in angle can be written as a sum of the required angles from each component. We saw that the  term was dominant in equation (60) and would cause inter-robot and obstacle collisions. We chose to shut off some of the inputs at strategically chosen moments. To avoid overwhelming the rotation inputs we shut off certain inputs depending on what state a given robot was in. We saw that the largest source of error was in avoiding obstacles so we decided to base our switching algorithm based on whether a robot detects or collides with and obstacle.

Table 2. Switching Conditions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Adjacent Robots () | Diagonal Robots () | Obstacle Avoid  () | Path to Target () |
| Not Detected | On | On | Off | On |
| Detected but No Collision | On | Off | On | Off |
| Detected and  Collision | Off | Off | On | Off |

Table 2 shows the conditions for when certain inputs are turned on or off. When an obstacle is not detected, a robot will to continue to update its orientation based on input from adjacent robots, diagonal robots and the path to its final target. When and obstacle is detected, the input to the diagonal robots is shut off to allow the system more flexibility to warp and avoid obstacles. When a collision occurs, all inputs are shut off except for obstacle avoidance.

To summarize, we can use inter robot distance , distance to obstacles and distance to target as input to the control system of each robot. If we have several distance sensors on the robot we can determine what angle ,the neighbor, obstacle or target is with respect to a robot’s heading. Using this information, we can calculate, the required rotation for robot to undertake which will point it in the desired heading. The desired heading will drive the robot towards satisfying the weighted consensus equation.

# Chapter 4: Simulation and Results

We developed our simulation using the Python programming language and used the *pygame* library to create a visualization. We also used the *math*, *numpy* and *random* libraries for the mathematical calculations involving matrix algebra, trigonometry and random number generation. Finally, we exported our data on the total energy and process time as a csv file to graph our data in Microsoft excel. The figures that show the system behavior were taken directly from the simulation visualization. We varied the number of robots in the system and settled on eight on run the main performance tests for Total Energy comparison. Anything less than a 6-robot system would not demonstrate the dynamic reconnection capabilities of our algorithm and large numbers of robots would take longer to converge.

## 4.1 Simulation controller

We will be comparing two system equations for our experiment. The first equation depends on the robot’s exact position relative to a fixed reference frame in the simulation. The equation of motion has the form:

|  |  |  |
| --- | --- | --- |
|  |  | (62) |

In equations (62), [xi, yi] is the position of the robot  with respect to the simulation reference frame.  is the set of robots connected to robot  as described by the Weighted Graph Laplacian Matrix, . The Graph Laplacian has prescribed Adjacency and Degree Matrices according to the desired formation and dynamic weights which are functions of inter-agent distances. Also,  the is  member of the set of all obstacles in range of robot as described by equation (40).

The second equation of motion we tested had the form:

|  |  |  |
| --- | --- | --- |
|  |  | (63) |

Where eachandare the positions in our simulation for all the agents in our system and is the orientation with respect to the simulation reference frame. We can update the orientation through the input from equation (61) which are the commanded angles based on our algorithm from section 3.6. We also use the prescribed values for  as described by equation (60) to determine the speed of each robot in the simulation. We consider the above equations as the control system for any physical or simulated robot with the ability to rotate and move. and are the inputs from the modified weighted consensus equation equations (60) and (61).

## 4.2 Test Cases

We submitted both control strategies to a series of tests and compared the behavior of the total energy with respect to process time for each:

1. *Rendezvous and achieve formation*

We initialized random positions for the agents and allowed them to converge and create a formation. We commanded the system to from regular polygons or circular formations for larger numbers of robots. We also showed examples of arbitrary shapes, in our case alphabetical letters, to demonstrate the generalization of algorithm to other shapes. We also looked at cases where the system settles in a false formation and how both algorithms deal with these situations.

1. *Maneuver the formation through a set of static obstacles.*

The system was initialized already in formation and allowed to navigate a static obstacle course. We performed our tests on eight robots and compared the behavior of the total energy with respect to the process time for both the absolute position algorithm and the robot frame algorithm.

1. *Maneuver the formation through a set of moving obstacles.*

The system was initialized in formation and could navigate through a dynamic obstacle course. The obstacles were initialized in a line connecting the formation and the target location. The obstacles could move randomly in the vertical direction but not move in the horizontal direction. The intended effect was to cause the formation to separate and deform.

## 4.3 Simulation Results

We will be using 8 robots for all our data collection when comparing the Absolute Position Algorithm (62) and the Robot Frame Algorithm (63). We chose 8 robots because it allows us to demonstrate the local minimum avoidance algorithm which requires gaps in the graph of system. We did not set out to test convergence relative to number of robots. Instead we are comparing convergence rate and total energy behavior relative to the two versions of the algorithms. The workspace is 800x800 cells and the cells are discrete meaning the position of a robot can never be between cells. The constant k for all the energy functions including  and were set to 500. In the simulation, it is assumed that robots and obstacles that are in between two robots or a robot and an obstacle, do not interfere with their connections. The prescribed inter-robot adjacent distance was set to 50 cells. We used the algorithm described by the section on regular polygon formation (3.3) to build the Weighted Graph Laplacian.

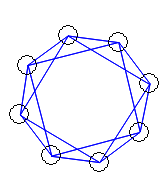
****

Figure 8. Formation to Regular Polygon with Eight Robots

Figure 8 shows the final formation to a regular polygon using the formation matrix definition described in Section 3.3. The final shapes for both Robot Frame and Absolute Position Algorithms were identical so we provided a single example in Figure 8.

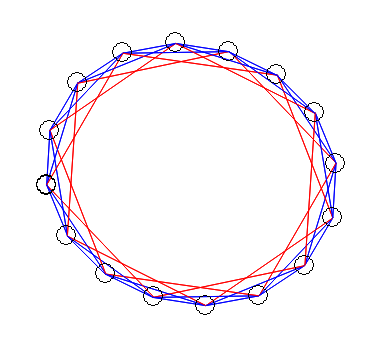


Figure 9. Formation to Regular Polygon with 16 Robots

Figure 9 shows the final formation to a regular polygon using the formation matrix definition described in Section 3. We used n = 16 robots to show that our method can be generalized to larger multi-robot systems. The red colored connections represent the previously unconnected vertices in the graph that are preventing the structure from collapsing into a false formation. These are activated when  as described in equation (38) in section 3.4 on False Formation avoidance.

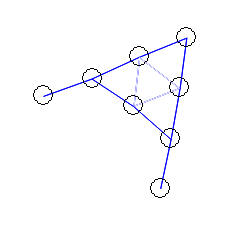
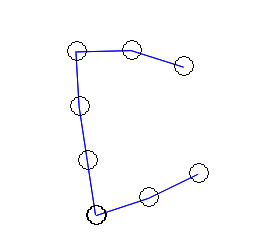
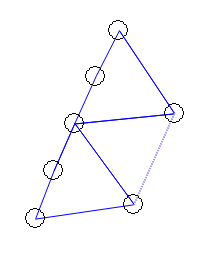
* *

Figure 10. Formation to Customized Formations

Figure 10 shows the final formation for explicit formations predefined by weights in the Laplacian Matrix. In this case we commanded the robots to form alphabetical letters. This shows that the method cam be generalized to any shape or formation. The formations require reinforcement connections which are blanked out during the screen refresh.

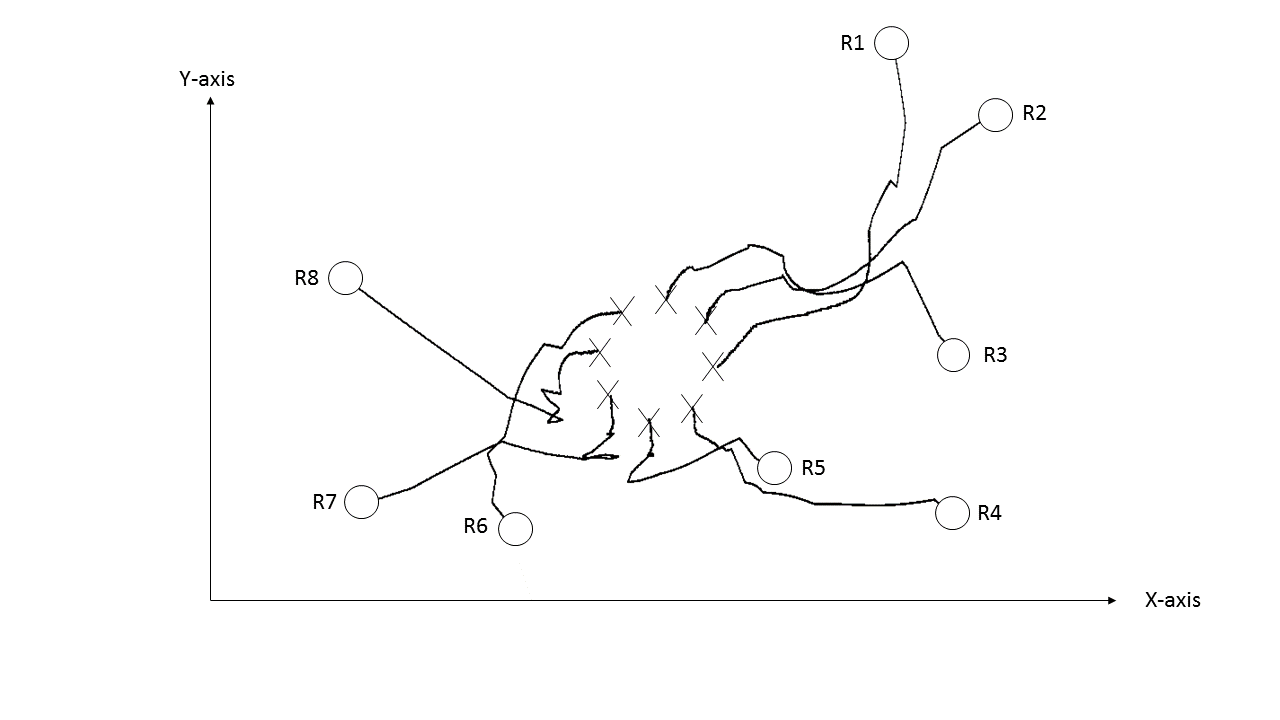
****

Figure 11. Convergence Paths

Figure 11 shows the path of all agents as they converge to a formation. In this case eights agents are forming a circle. The agents are initialized randomly and are immediately connected to their neighbors as described in the Formation Matrices Section in Chapter 3.

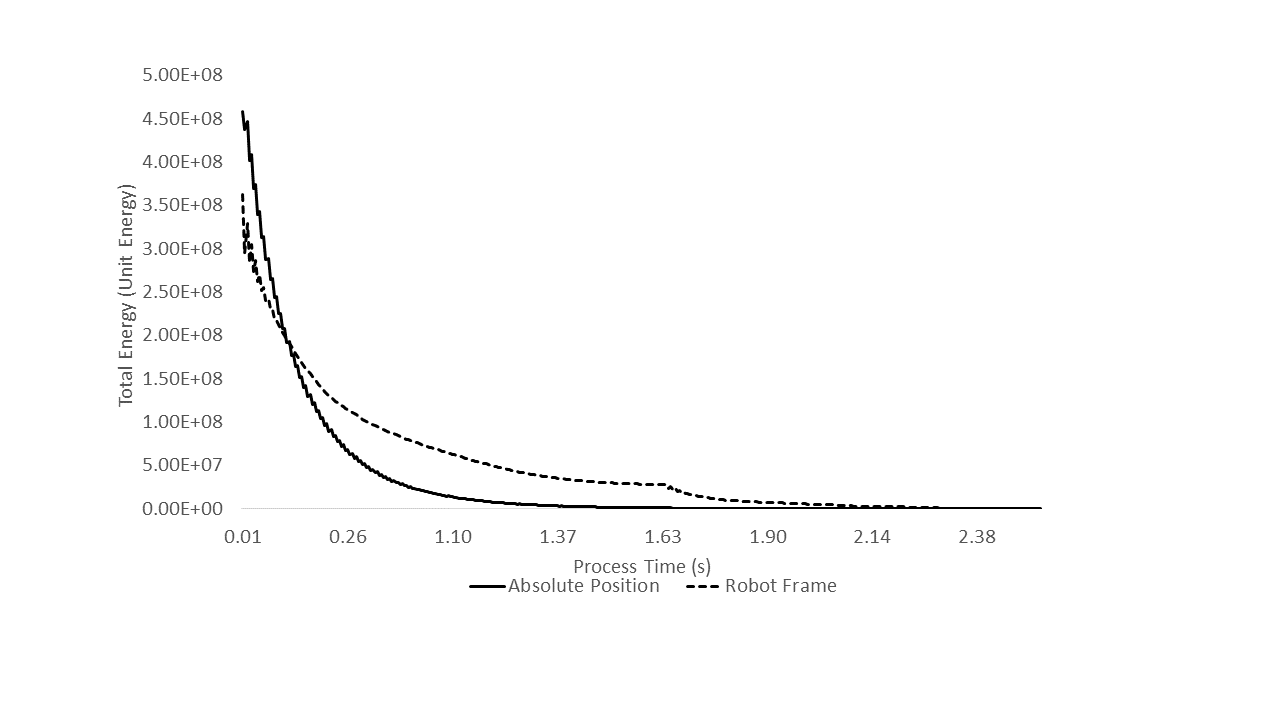


Figure 12. Total Energy during Convergence

Figure 12 shows the total energy of both the Absolute Position and robot Frame algorithms as the system converges to a formation. The horizontal axis gives the CPU time in seconds for the system to converge to a formation. The simulation used 8 robots and their positions were initialized randomly. The agent positions are initialized randomly and allowed to move according to the equations (63) and (64) for the respective Absolute Position and Robot Frame algorithms. Both the fixed frame and robot frame based algorithms total energy profile exhibited behavior like that shown in the figure 12. The gradient descent algorithm converges asymptotically so we had to make a cutoff point to end the simulation. We made the cut off point for ending the simulation to be 1000 units of energy. At 1000 units of energy we define that the system has reached its final form. This is based on visual inspection and trial and error from the simulation. In general, larger systems consisting of more robots took longer to converge to formation. The absolute position algorithm approached lower energies faster but took longer to converge to 1000 when compared to the Robot Frame Algorithm.

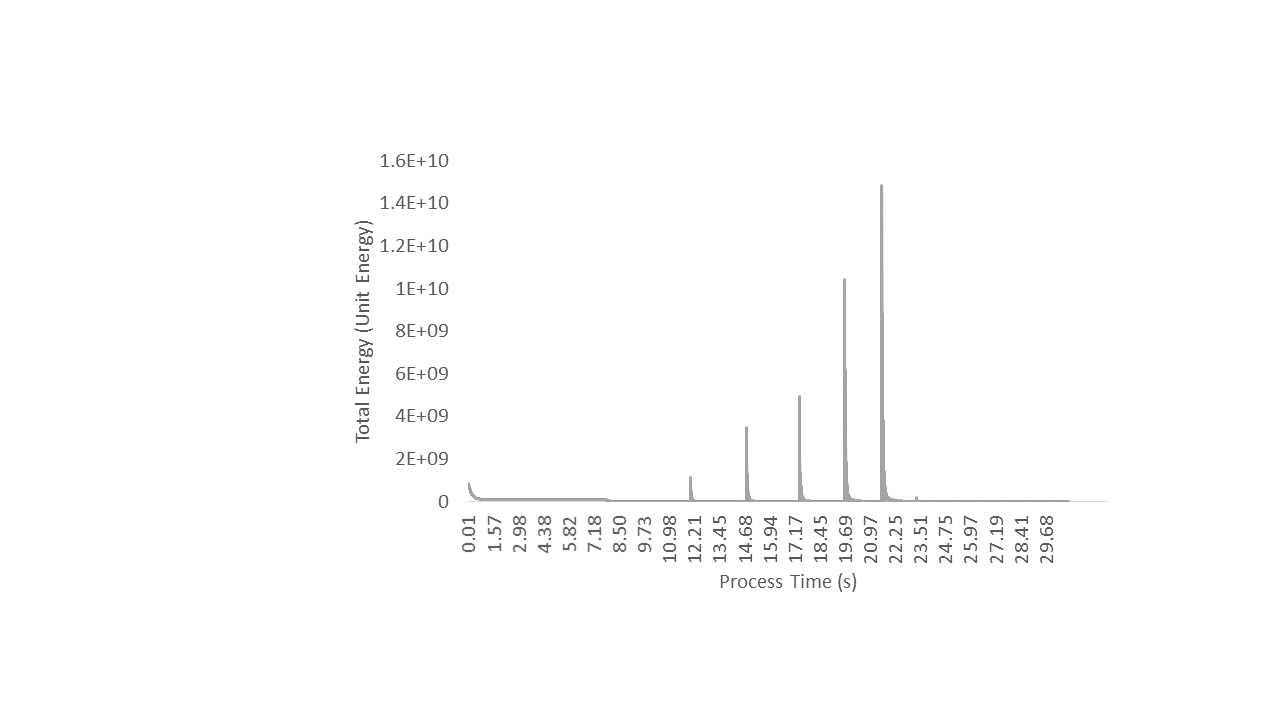
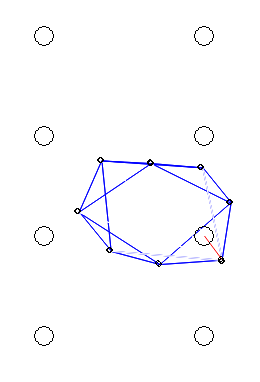
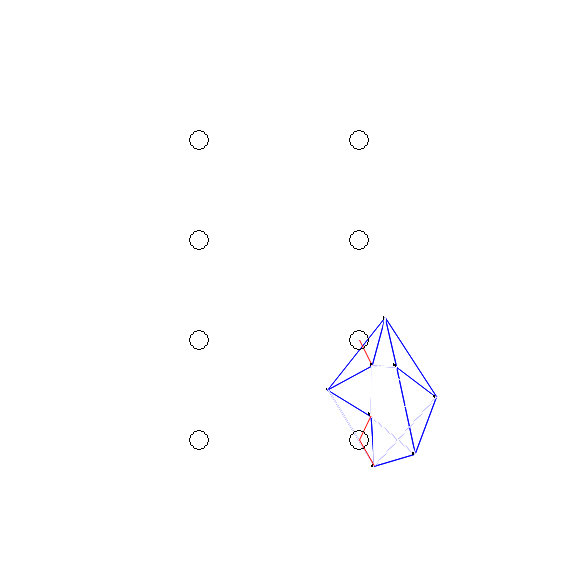
****

Figure 13. Total Energy during False Formation Avoidance

Figure 13 shows the total energy as the system converges to the formation in figure 8 and avoids a false formation. We used the absolute position algorithm as a demonstration. The spikes in energy are caused by the temporary connection between previously unconnected vertices in the system graph. Temporary connections in the graph resets the convergence procedure so that it may try again. This behavior is defined by the dynamics of as described by the algorithm in Table 2 and the weight function in equation (38).

For the False Formation avoidance algorithm, we chose the following parameters:  ,and . The parameters were chosen through trial based on worked in our simulation. The Absolute Position Algorithm always required several iterations though the Local Minimum Avoidance procedure to achieve the global minimum while the Robot Frame Algorithm typically only required one iteration. The reasons for this is due to the constant motion of the agents in the Robot Frame Algorithm. The greater variance in speed randomized the positions of the robots more so than the Absolute Position Algorithm. The ABP slowed down as it came to a false formation and so there was a smaller probability to escape from the false formation initially. The number of cycles through the algorithm is characterized by the number of spikes in the energy graph.

We then simulated the 8-robot system maneuvering through a set of static obstacles. They began to move through the obstacle course after already achieving the desired configuration. The size of the obstacles is not considered as both robots and obstacles are modeled as points. If an obstacle gets between two connected robots, the connection is not interrupted. For both the Absolute Position Algorithm and Robot Frame Algorithm, the robots are unaware of the obstacles they come in range of predefined distance as described equation (40) in section 3.5.

** **

1. (b)

Figure 14. Maneuvering Around Static Obstacles

Figure 14 shows both algorithms navigate through two rows of static obstacles. The figure on the left (a) the shows the visualization for the absolute position based algorithm while the figure on the right (b) shows the visualization for the robot frame algorithm. The range of detection for obstacles was specified by the value of as defined in equation (40) in section 3.5 on obstacle avoidance. The value chosen in the code was 30 cells or three times the size of the virtual robot radius. The robot radius has no physical meaning since the robots are modeled as dimensionless points. This value can be compared to the prescribed inter-robot adjacent distance of 50 cells. The value chosen for was large enough that it could avoid obstacles and was found through trial and error.

We can see that the robot frame algorithm has a harder time staying in formation when compared to the absolute position algorithm. The reason for this is slower response time of the Robot Frame algorithm due to it having more lines of code. An example of the code is shown at the end of this chapter.

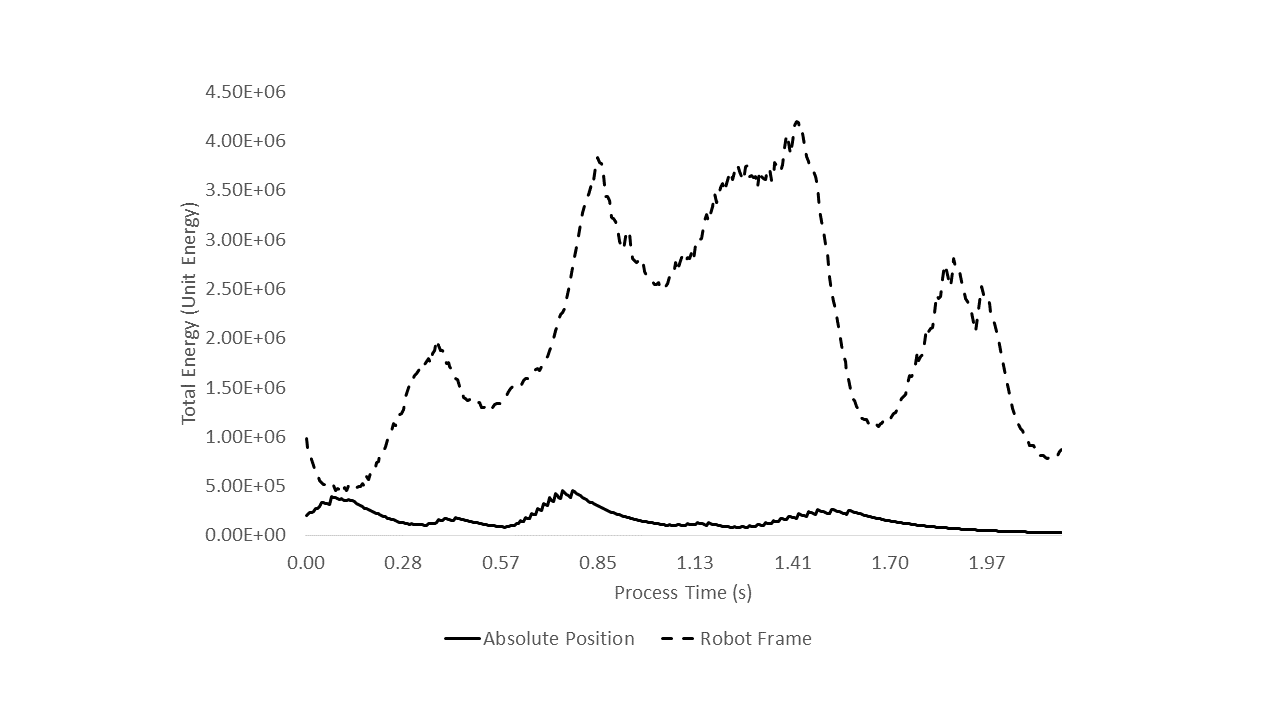
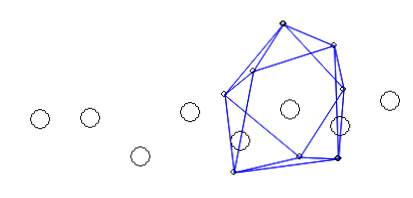
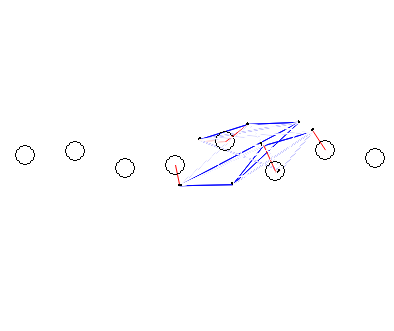
****

Figure 15. Total Energy during Static Obstacle Avoidance

Figure 15 shows the total energy of both algorithms as the system navigates through a set of eight static obstacles. Since the robot frame algorithm has a harder time staying in formation as shown in Figure 4.7 the total energy increases as the system spreads apart.

**

1. (b)

Figure 16. Maneuvering Around Dynamic Obstacles

Figure 16 shows both algorithms navigate through a row of moving eight obstacles. The figure on the left (a) shows the visualization for the absolute position based algorithm while the figure on the right (b) shows the visualization for the robot frame based algorithm. The obstacles move vertically in random directions are constrained horizontally. The range of detection is again chosen to be 30 cells. This can be compared to the prescribed inter-robot adjacent distance of 50 cells. The obstacles appear to be in different configurations because the visualizations were captured at different times and they move in random vertical directions during every simulation. Again, the robot frame algorithm deforms more easily.

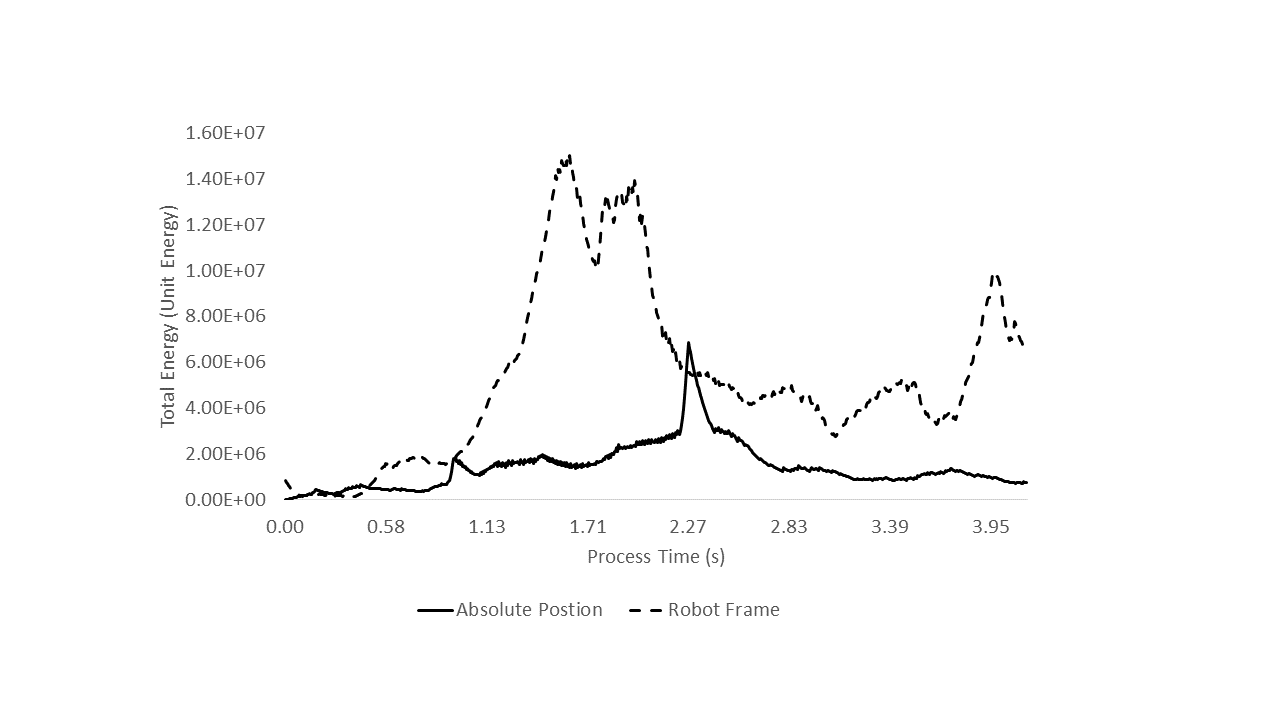
**

Figure 17. Total Energy during Dynamic Obstacle Avoidance

Figure 17 shows the total energy of both absolute position and robot algorithms as the systems navigate through a set of eight moving obstacles. The total energy for the robot frame algorithm increases more dramatically than it did for the static obstacle course and does not recover by the time it reaches the end of the obstacle course.

We can see that the Absolute Position Algorithm was more likely to maintain formation while maneuvering through both static and dynamic obstacles. This can be seen in both the visualizations and in higher energy spikes in the energy graph. The divergence in total energy can be seen more dramatically in the Moving Obstacle Test. The energy of the Robot Frame algorithm continues to diverge throughout the test while the Absolute Position Algorithm total energy drops to a lower level. The divergence in energy for the robot frame algorithm is caused by a delay as the robots attempt to update their commanded headings based on sensor input. The absolute position algorithm updates the commands in fewer lines of code and can therefore do so more rapidly.

|  |
| --- |
| # Robot Position is Updated  self.Pn[:, i] = self.Pn[:, i] + (self.P[:, j] - self.P[:, i]) \*0.00001 \* W[i,j] + self.move \* (self.target[:] - self.P[:,self.minimum]) \* 0.0005 + (self.O[:,j] - self.Pn[:,i])\* w\_obj[i,j] \*0.001 |

|  |
| --- |
| # Robot to Target Calculated  self.phi\_target[i] = math.atan2(self.target[1] - self.P[1,i] , self.target[0] - self.P[0,i]) - self.Rn[i] self.U\_target[:,i] = [np.linalg.norm(self.target[:] - self.Pn[:, i] ) \* math.cos(self.phi\_target[i])\*v\_target[i],  np.linalg.norm(self.target[:] - self.Pn[:, i] ) \* math.sin(self.phi\_target[i])\*v\_target[i]]  #============================================================================================ # Robot to Obstacle Angle Calculated self.phi\_obj[i,j] = math.atan2(self.O[1,j] - self.P[1,i] , self.O[0,j] - self.P[0,i]) - self.Rn[i] self.U\_obj[:,i] = [np.linalg.norm(self.O[:, j] - self.Pn[:, i] ) \* math.cos(self.phi\_obj[i,j]) \* w\_obj[i,j],  np.linalg.norm(self.O[:, j] - self.Pn[:, i] ) \* math.sin(self.phi\_obj[i,j]) \* w\_obj[i,j]]  #============================================================================================ # Inter Robot i's angle input self.phi[i,j] = math.atan2(self.P[1,j] - self.P[1,i], self.P[0,j] - self.P[0,i]) - self.Rn[i]  self.phi[i,i] = 0  # Inter robot Rotation is calculated self.U\_bot[:, i] = [np.linalg.norm(self.Pn[:, i] - self.Pn[:, j]) \* math.cos(self.phi[i,j])\*W[i,j]  ,np.linalg.norm(self.Pn[:, i] - self.Pn[:, j]) \* math.sin(self.phi[i,j])\*W[i,j]]  #Angels are Updated self.Rn[i] = self.Rn[i] + math.atan2(self.U\_bot[1, i], self.U\_bot[0, i]) + math.atan2(self.U\_target[1, i], self.U\_target[0, i]) \* self.move \* 0.5 + math.atan2(self.U\_obj[1, i], self.U\_obj[0, i])  # Speeds are calculated based on the energy of neighbors self.s[i] = V[i].sum() + v\_obj[i].sum() + v\_target[i] \* self.move \* 0.5  #Speeds are capped. **if** self.s[i] > 40000:  self.s[i] = 40000  # Bot positions are updated. self.Pn[0, i] = self.Pn[0, i] + self.s[i] \* math.cos(self.Rn[i]) \* 0.00001 self.Pn[1, i] = self.Pn[1, i] + self.s[i] \* math.sin(self.Rn[i]) \* 0.00001 |

Furthermore, we deactivate certain sensor inputs for the robot frame algorithm based on proximity to obstacles which further delays the commanded headings and speed for our algorithm. The advantage of the robot frame algorithm is that it does not depend on knowing the position of each robot with respect to a stationary frame. The robot frame algorithm only requires relative distance to neighboring robots and their direction with respect to a robot fixed frame. This allows for a more decentralized control strategy then is allowed with the Absolute Position Algorithm.

## 4.4 Comparison to Previous Implementations

Both the Absolute Position and Robot Frame Algorithm were derived from our research into our main reference papers as described by our Literature Review in Chapter 2. In our primary reference paper, titled *Edge Weighted Consensus Based Formation Control Strategy with Collision Avoidance* [1], the authors implemented an algorithm like ours with some key differences. The differences are as follows:

1. *Energy Function*

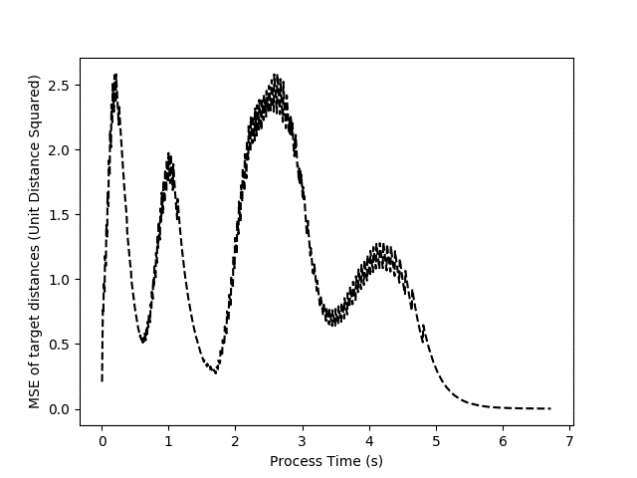
The paper uses an energy function of the form of a hyperbolic cotangent function which gives an asymptote near small inter robot distances. We chose to use a parabolic function because of its ease of implementation and to experiment with a function that did not have an asymptote.

1. *False Formation Avoidance Algorithm*

The reference paper uses a virtual relabeling algorithm described in Chapter 5 of the reference paper. The algorithm calculates desired positions with respect to the centroid of the system. When the energy function approaches a *local minimum,* as defined by the paper, the index label for each robot is reassigned continuously until the global minimum of the total energy function is reached.

We compared the performance of our best performing algorithm, that being the Absolute Position version, with the performance of the implantation from the reference paper. They provide data on the mean square error between the desired inter robot distances and the actual distances for a system composed of 6 robots as it moves through a set of static obstacles. They also provide data for the Error in distance from the centroid of the system to all the robots in a three-robot system with a communication delay of 0.5 seconds. We prepared our simulation for both test cases and collected data on the same performance metrics under the same conditions for the absolute position algorithm. The simulation was created in an arena with a size 8x8 unit distances (u). A unit distance is an arbitrarily scaled distance which allows us to see the behavior of the system. Our results are shown in figures 17 and 18. The numerical comparison to the reference results are shown in table 3.

Figure 17 shows the Mean Square Error for inter robot distances for a 6-robot system with K = 500 as it navigates through a set of static obstacles. We compared the results from the reference paper which are shown in section 6.2, page 18 of [1]. The spikes in the graph were caused by the system encountering obstacles and becoming deformed as individuals attempted to avoid them. The peak deviation for Figure 17 was 2.5 u2. For K = 5000 the peak deviation for our simulation was 1.65 u2. We increased the size of our constant in the energy function by a factor of 10 as it was in the reference paper.



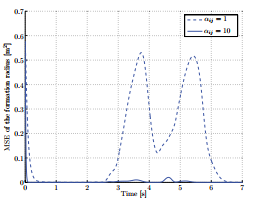


Figure 18. Mean Square Error for Inter-Robot Distances with K = 500

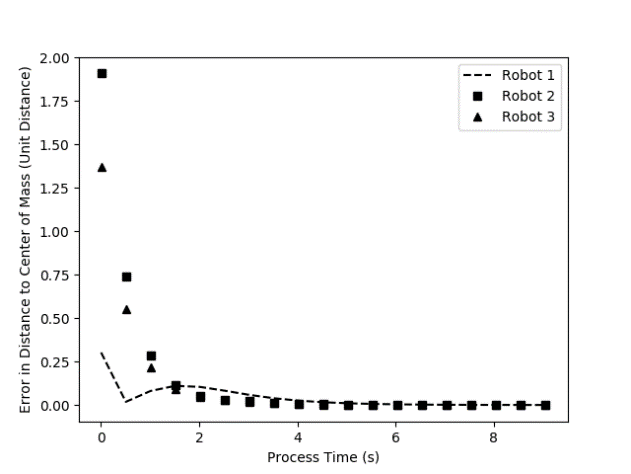
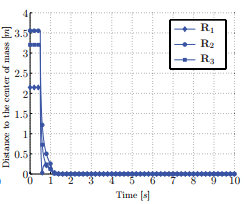


Figure 19. Error in Distances to Center of Mass

A communication delay of 0.5 seconds was introduced to our simulation to replicate the test conditions in section 6.3, page 19 of [1]. The results are show in Figure 18. Three robots were initialized randomly and allowed to converge to a formation with equal inter-robot distances, creating an equilateral triangle. The reference paper measured the error between the desired distance and actual distance to the centroid for each robot. We scaled our distance measurement to arbitrary units to show the shape of the graph as shown in the experimental results section of the reference paper.



|  |  |  |  |
| --- | --- | --- | --- |
|  | Convergence Time (s) | Peak Deviation for Small constant (u2) | Peak Deviation for Large constant (u2) |
| Reference [1] Algorithm | 10.0 | 0.52 | 0.02 |
| Absolute Position Algorithm | 9.0 | 2.51 | 1.65 |

Table 3. Absolute Position and Reference Algorithm Comparison

Figure 18 can be compared to the results of [1]. The behavior of the 3 robots moving towards convergence are qualitatively similar in both implementations. The time to convergence for our case is about 9 seconds while the reference paper’s system converges in about 10 seconds. Another similarity is the initial drop in error for Robot 1 in our simulation and Robot 2 in the reference experiment. The large deviations for our Absolute Position Algorithm should not be compared to reference results as a measure of performance. Instead the data is provided to show the similarity between the two algorithms in a qualitative sense.

# Chapter 5: Conclusions

This thesis focused on a graph theoretic approach to multi-robot formation control. We compared an absolute position based algorithm previously developed by others, and an extension to the algorithm which removes dependence on an absolute reference frame. The extension to the algorithm relied on distance information between robots as well as the direction of their neighbors, relative to a given robot’s heading. The system was represented as a Graph and it was shown how the Graph Laplacian Matrix can describe important information about a multi-robot system. The edges of the graph were assigned a potential energy function and the total energy of the system was minimized by formulating the system equation as a gradient descent. We also used the graph structure to specify robot formation and demonstrated the relationship between the Graph Laplacian Matrix and specific robot formations. The experimental results show the limitations of the Robot Frame Algorithm mainly in its difficulty in maintaining formation while navigating through both a static and dynamic obstacle. The advantage of the Robot Frame Algorithm is its distributed structure and its independence from absolute position tracking.

Future works will aim to implement the Absolute Position Algorithm on physical robots to compare our results to behavior in the real world. We will also improve the collision avoidance scheme in the Robot Frame Algorithm. This will allow us to implement the algorithm on physical hardware as well. Finally, we would like to include connection enforcement techniques as described in the conclusions section for [1]. This will allow us to dynamically enforce connectivity in the communication graph.

# REFERENCES

[1] Falconi, Riccardo, Lorenzo Sabattini, Cristian Secchi, Cesare Fantuzzi and Claudio Melchiorri. Edge-weighted consensus-based formation control strategy with collision avoidance. Robotica, 33: 332-347, December 2013

[2] M. Zavlanos, M. Egerstedt, and G. Pappas. Graph Theoretic Connectivity Control of Mobile Robot Networks. Proceedings of the IEEE. March 2011.

[3] Kan Z, Klotz JR, Shea JM, Doucette EA, Dixon WE. Decentralized Rendezvous of Nonholonomic Robots with Sensing and Connectivity Constraints. The American Society of Mechanical Engineers. Journal of Dyn., Sys., Meas., and Control. 139(2): November 2015

[4] Nicholas Hoff, Robert Wood, Radhika Nagpal. Distributed Colony-Level Algorithm Switching for Robot Foraging. In Distributed Autonomous Robotic Systems 9: No Date

[5] Hartono P., Kakita S. (2008) Embedded Neural Network for Swarm Learning of Physical Robots. Internet Corporation for Assigned Names and Numbers: 18th international conference on Artificial Neural Networks, Part II, 141-149 : September 2008.

[6] Michael M. Zavlanos. Herbert G. Tanner, George J. Pappas. Hybrid Control for Connectivity Preserving Flocking, IEEE Transactions on Automatic Control: September 2009

[7] R. Pedrami, S. Wijenddra, J. Baxter, B.W. Gordo. A Control Allocation Approach for Energetic Swarm Control. IEEE International Conference: Robotics and Biomimetics, June 2009

[8] John Duchi, Elad Hazan, Yoram Singe. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. Journal of Machine Learning Research, Volume 12, July 2011, pages 2121--2159. Originally UC Berkeley EECS Technical Report 2010-24.

[9] FAX J. Alexander , MURRAY Richard M. Information Flow and Cooperative Control of Vehicle Formations. IEEE transactions on automatic control , vol. 49, no 9, pp. 1421-1603 [Document : 12 p.] (32 ref.), pp. 1465-1476 , 2004

[10] Aleksander Jevti´c, Diego Andina Swarm Intelligence and Its Applications in Swarm Robotics. CIMMACS'07 Proceedings of the 6th WSEAS international conference on Computational intelligence, man-machine systems and cybernetics, Pages 41-46, 2007

[12] Yichen Yu. Swarm Intelligence. Student Essay on Swarm Intelligence: 2012 guava.physics.uiuc.edu/~nigel/courses/569/Essays\_Fall2012/Files/Hu.pdf

[13] Yogesawaran Mohan, S.G. Ponnambalam. An Extensive Review of Research in Swarm Robotics. In Nature Biologically Inspired Computing: January 2010.

[14] Giovanni Pini, Matteo Gagliolo, Arne Brutschy, Marco Dorigo, Mauro Birattari. Task partitioning in a robot swarm: a study on the effect of communication. Swarm Intell (2013) 7: 173. doi:10.1007/s11721-013-0078-7

[15] https://www.algebra.com/algebra/homework/Polygons/Polygons.faq.question.785381.html

# APPENDIX