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: HelenYu, 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**

There are a variety of methods proposed for establishing a distributed controller for a multi-robot system. This thesis will focus on a graph theoretic approach to multi-robot formation control. We will compare an absolute position based algorithm previously developed by others, and an extension to the algorithm which removes dependence on an absolute reference frame. These robots will have access to distance information between robots as well as the direction of the neighboring robots relative to a given robot’s heading. The system will be represented as Graph and it will be shown how it can be described by the Graph Laplacian Matrix. The edges of the graph will be assigned a potential energy function and the total energy of the system will be reduced by formulating the system equation in the form of a gradient descent. We will also use the graph structure to specify robot formation and demonstrate the relationship between the Graph Laplacian Matrix and robot formations.

TABLE OF CONTENTS

Page

[LIST OF FIGURES 5](#_Toc439838672).

[LIST OF TABLES 6](#_Toc439838672).

[Chapter 1. Introduction 7](#_Toc439838673).

[Chapter 2. Literature review 10](#_Toc439838674).

[Chapter 3. Background 13](#_Toc439838683).

3.1 Graph Theory13.

3.2 Edge Tension Energy Minimization17.

3.3 Formation Matrices22.

3.4 Avoiding Local Minima24.

3.5 Obstacle Avoidance26.

3.6 Removing Dependence on Absolute Position 28.

[Chapter 4. Simulation and Results 35](#_Toc439838702).

4.1 Simulation Controller35.

4.2 Test Cases36.

4.3 Simulation Results37.

[Chapter 5. Conclusions and future work 42](#_Toc439838721).

[REFERENCES ?](#_Toc439838722).

[APPENDICES](#_Toc439838723)

[Appendix A Python codes ?](#_Toc439838724).

# LIST OF FIGURES

Figure Page

Figure 3.1 Graph with Inter-Agent Tension Enrgy 16.

Figure 3.2 Energy Function 17.

Figure 3.3 Formation Graph 21.

Figure 3.4 Obstacle Avoidance 30.

Figure 3.5 Agent Network with Stationary Frame 25.

Figure 3.6 Robot Network with Agent Frame 28.

Figure 4.1 Robot Network with Agent Frame 28.

Figure 4.2 Robot Network with Agent Frame 28.

Figure 4.3 Robot Network with Agent Frame 28.

Figure 4.4 Robot Network with Agent Frame 28.

Figure 4.5 Robot Network with Agent Frame 28.

# LIST OF TABLES

Table Page

Table 1 Local Minimum Avoidance 16.

Table 2 Switching Conditions 17.

Table 3 Collision Events 21.

**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 Mars 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 mission failure if one of the members of the system become inoperable. Another valuable application of swarm intelligence in robotics is in search and rescue. Small robots, with swarming capabilities can search destroyed building for survivors by being able to reach places that are unreachable by 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 correction operation of the telescope.

This thesis will study robotic swarm motion coordination and how robots within a swarm calculate and readjust their positions with respect to each other using a Particle Swarm Optimization(PSO) algorithm. The rendezvous problem will be modeled and solved using PSO and modifications will be introduced to allow for greater control over a particle swarm. The goal of this project is to develop a particle swarm optimization algorithm, which allows for semi-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.

We will start in chapter 3 by determining the distances between all the individuals and update their positions iteratively by subtracting from their current positions. The end effect would be the convergence of the swarm on one point.  However, this is unrealistic for a physical robotic swarm because if would cause a collision at the convergence point.  We can introduce weights to the algorithm to slow down and control the rate of convergence. Once we have established a way to maintain a constant inter-agent system, we will introduce an extension to strategy that includes obstacle avoidance. We also differentiate our work from our references by introducing a transformation of frames so as state our system equation indecent of a fixed frame.

In the Chapter 4, we will introduce test cases and compare the performance of the original control system developed in [1] and [2] and compare it to our frame independent control strategy. We will define a set of objectives the systems should accomplish including:

1. Rendezvous and achieve formation
2. Rendezvous and achieve formation with obstacles in the way.
3. Maneuver the formation through a set of static obstacles.
4. Maneuver the formation through a set of dynamic obstacles.

Finally, in Chapter 5 we will present our conclusions and describe our intentions with the continuation of the project, including the implementation of the algorithm on hardware.

**Chapter 2: Literature Review**

We found several Graph Theoretic Methods for establishing multi-agent control systems as well state switching algorithms such as [4] as well as swarming method based on real time learning using an embedded neural network [5]. Since our method is based on Graph Theory we reference those paper heavily while also comparing them to alternative methods.

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 Matrix 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 3.4.

Our definition for the Graph Laplacian is found in [2]. The paper also provides a weighted version of the graph Laplacian as described in this thesis and introduces a theoretical framework for controlling graph connectivity in mobile robot networks. [2] uses a finally trigonometric function as their energy weights instead of our parabolic function. We chose a parabolic function because of its ease of implementation and to avoid having to fine tune. They derived their weights functions in the exact same as we did however. [2] also introduces an obstacle avoidance augmentation of their algorithm which is useful for guiding the formation around and through a space filled external objects. I have done some exploration into their method have not tested fully as of (March 10th 2017).

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 attempt is different from the attempt in this thesis because they do not invoke Graph Theory and instead derive their algorithm from dynamics. This will be used as an example for comparing our method to an alternative approach.

This paper [4] focuses on foraging as a multi-robot task and present 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 similar to 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 have already implemented algorithm similar to one described in [4], it is possible we might be able to merge the task of resource foraging using the formation control derived in this thesis.

In this paper [5] real time learning of multiple physical autonomous robots situated in a real dynamic environment is 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 are allowed to run until the desired inter-robot distances are achieved. This method will be compared to our approach to compare convergence rate and complexity.

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. So far (March 10th 2017) our algorithm assumes connectivity and has pre-supposed link deletions based on the formation we would like to achieve prior to starting the simulation. If we time allows it we will explore the method in [6] further and attempt to integrate it into our algorithm.

This paper [7] invokes 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 algorithm and interfacing it with a low level controller for differential drive robots. This paper [8] is a Master’s thesis by a student from at Delft University of Technology. The paper discusses the control of robot swarms using Radio Signal Strength measurements on a group of six legged robots. The method used is not Graph Theoretic but is derived from the dynamics of the robots themselves. I am using [8] as a model for my own thesis and may not be included in the final literature review.

**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.



 is the Vertex set,

 is the Edge set.

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 defineto be the set of vertices that are connected to 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.







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 

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.

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



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



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]. The graph Laplacian has few special properties [1]:

1. L is positive semi-definite
2. The eigen-values of the Laplacian Graph are always non-negative. They can always be ordered as:



1. If is a simple eigenvalue (i.e. ), then the graph is connected. In this case,



Where and are vectors of N elements all equal to 1 and 0 respectively. This implies L1 = 0.

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



In our system,is the position of an agent and n is the total number of agents. 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 single point.





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:



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:



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.



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



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



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 approach and that 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].

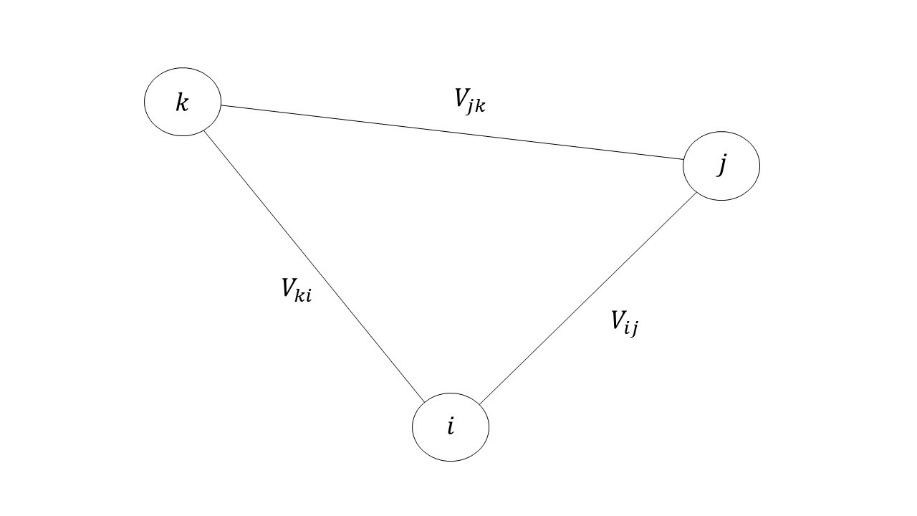


The weights are functions of the Euclidean norm (distance) of the positions between two adjacent agents. The distance is defined as:



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 3.1 Graph with Inter-Agent Tension Energy



*Figure 3.1. 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:



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



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



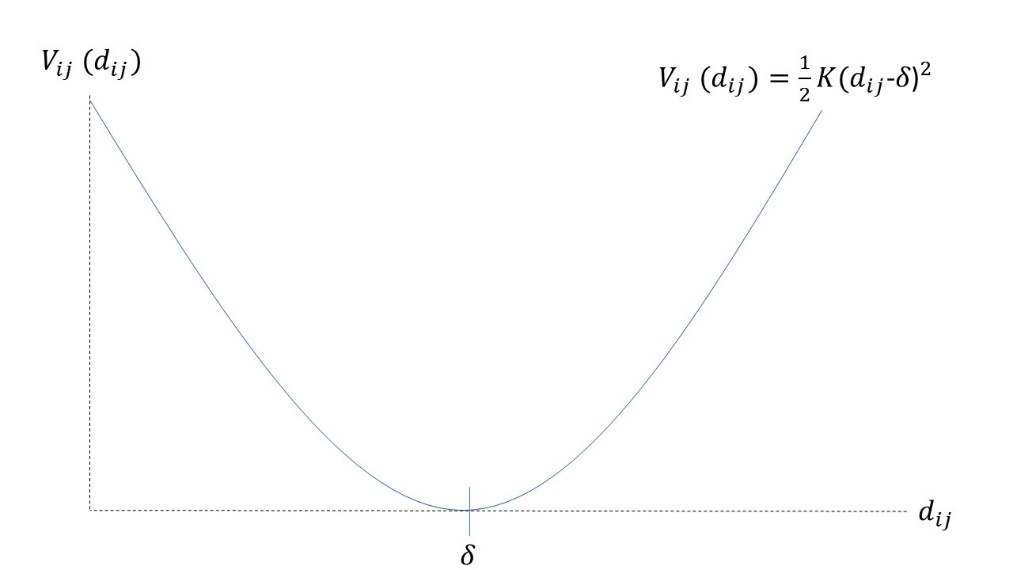
The desired outcome is achieved because the derivative of the total energy with respect to time is always negative and therefore the energy is always decreasing.



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:



Figure 3.2 Tension Energy Function



*Figure 3.2 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:



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



Therefore, the system equation can be rewritten as:



The weight functions can be written as:



By writing the control algorithm as a gradient descent, two agents moving away from each other will undergo an attractive 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].



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







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



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



The above equation describes the behavior of the multi-robot system where each robot has an equation of motion in the form of:



And the total system is guided by the gradient descent algorithm:

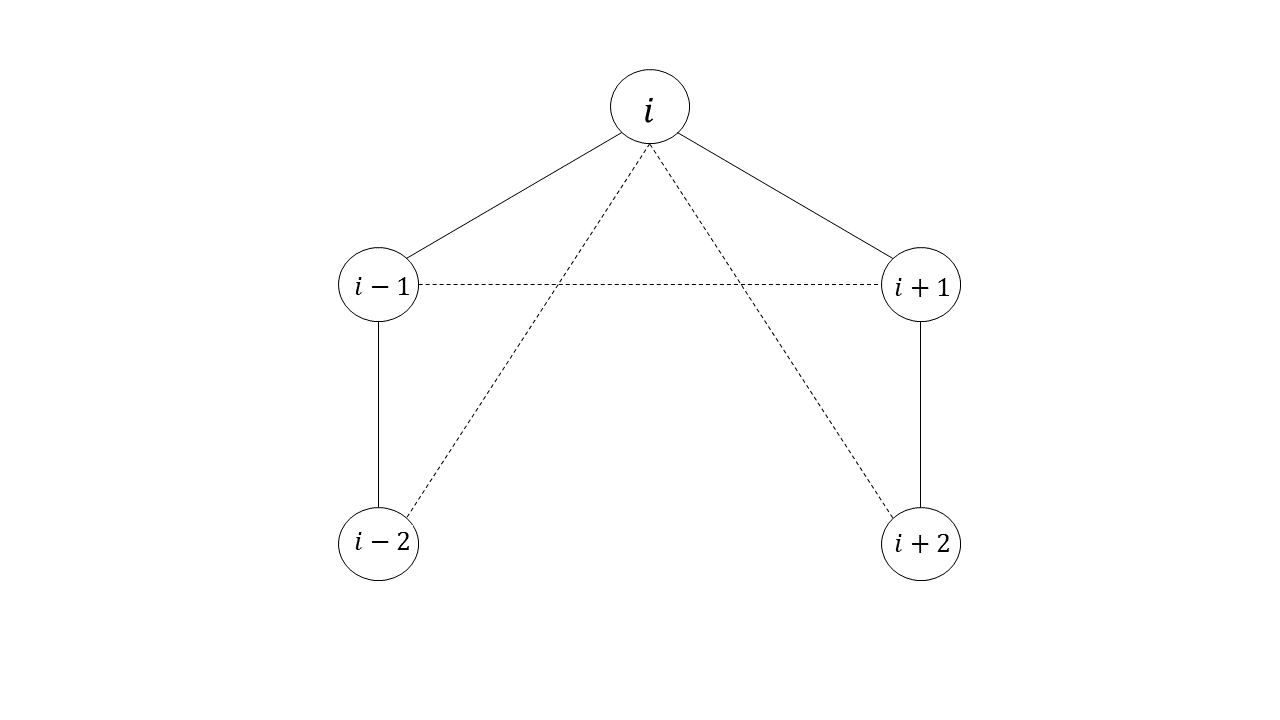


Where the total energy is minimized when the inter-robot distances are equal to the prescribed distance . This allows us to space the robots apart and maintain that distance under perturbations. However, we can see that not all robots can be the same distances from all other robots simultaneously which 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 Graph Laplacian Matrices that define our target formations. In the following section, we will describe how we can choose values for the Adjacency and Degree matrix to create certain unique shapes with the multi-robot system.

***3.3 Formation Matrices***

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. We will proceed to build regular polygons with different number of vertices. To avoid fix formatting the values of the graph into system and allow for a more dynamic behavior within 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 3.3 Formation Graph



*Figure 3.3 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 unit length. In the figure 3.3, these vertices have index and relative to a given vertex. When defining our Weighted Adjacency Matrix the input for the weights will be . The non-adjacent connected vertices are known as the diagonal vertices and have indices andin figure 3.3 relative to a given vertex. The required distance for the diagonal edges can be found by the formula:



In this equation, n is the number of vertices in the graph. We can now define an algorithm for defining our weight functions for any value of n. If the weights in our system equation has the general form:



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. 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 graph that will need to be accounted for when the system becomes tangled or falls into a local minimum.

***3.4 Avoiding Local Minima***

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. For larger collections of robots, all the inter-robot target distances could be achieved 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.

To compensate for this, we establish temporary connections between robots that are not already connected by the 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 so that it creates a repulsive force below that activation distance but does not attract beyond that activation distance.

 If and

 If 

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. Furthermore, we initialized the activation distance to some arbitrarily large number. In our simulation, we set where n is the number of robots and is the side length of the target polygonal formation.

Throughout the formation processes we want the energy to decrease. If for any reason the energy begins to increase initially it means that is too large. While this is the case, b begins to increment, reducing the size of , and a counter c begins to increment. In the best-case scenario, is reduced to the point where it is no longer active, the total energy begins resumes its decline and reaches a global minimum. If the total energy has not begun to decrease by the time the counter c reaches its max value, it means that the system has fallen into a local minimum. In this case, b is set to 1, c is reset to 0 and a is incremented. This increases the size of with the intention of restarting the formation the process. This algorithm will repeat until a reaches its maximum value and the whole cycle repeats. The algorithm can be summarized with the following pseudo-code:

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

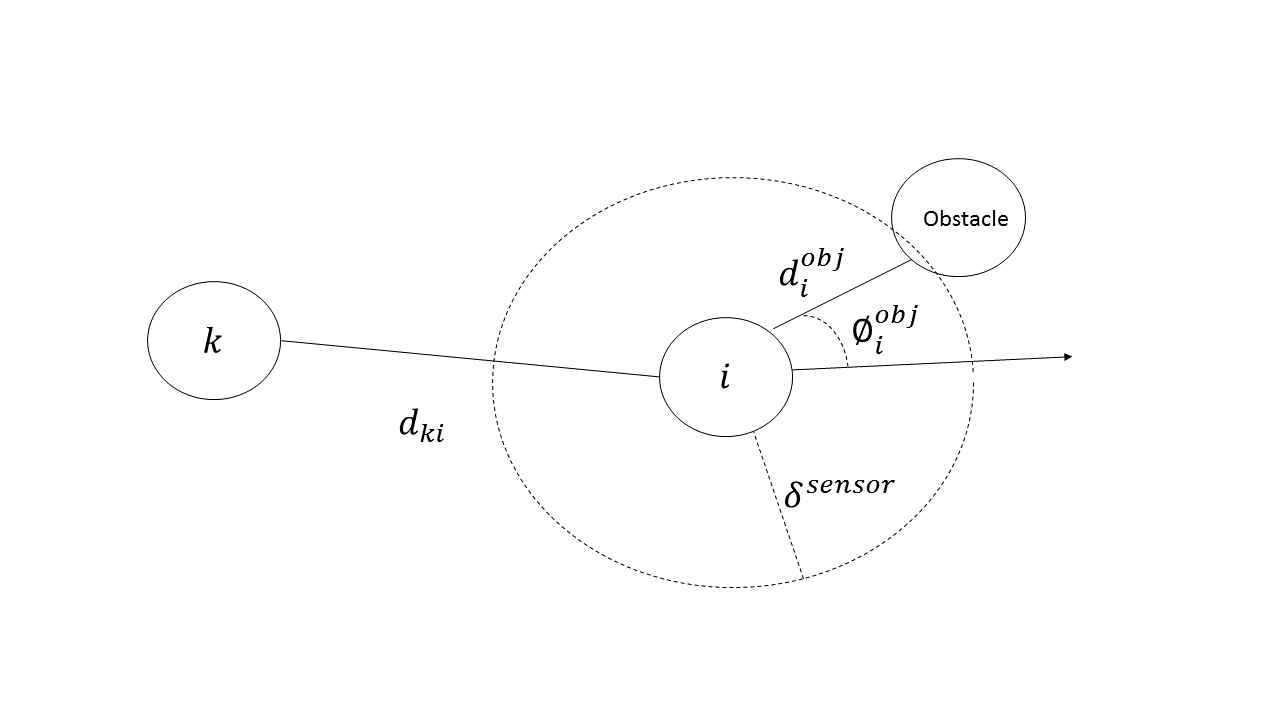
*Table 1 Pseudocode for Local Minima avoidance*

Evetime nonconnected agent get closer to each other than the activation distance, that distance is adjusted. The activation distance is capped so that it does not increase indefinitely and cause the system to diverge. The results of this technique are described in Chapter 4.

**3.5 Obstacle Avoidance**

Now that we have a method for achieve 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 3.4 Obstacle Avoidance



*Figure 3.4 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:



In figure 3.6, is the distance from the robot to the object and is the range of the sensor. The weights are activated when the obstacle is in the range of the sensor:

 If and

 If 

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 the robot’s sensor but will be zero or non-attractive when it is beyond the range of the sensors.

The weight terms are summed over the number of objects in range of robot .The consensus equation will now have the form:



In the equation above, is the position of the detected obstacle and is the set of obstacle vertices that are connected to the vertex . It is also trivial to add a term to the consensus equation that draws the system towards a target location. The resulting final form of the system equation is:

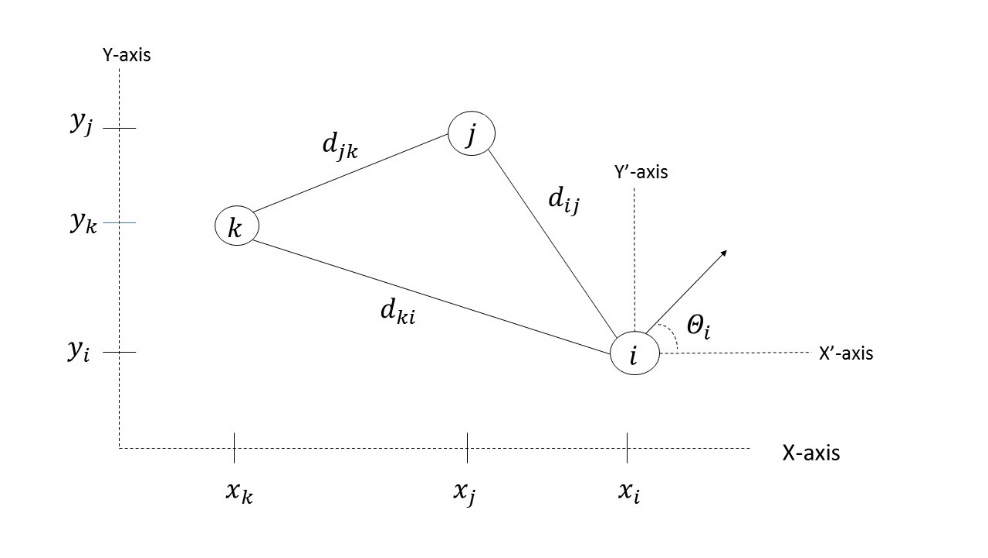


Where is the location of the target point for the whole system to move towards. We have now derived a total system equation of motion that can maintain formation, avoid obstacles and move towards a target location. The limitation of this strategy is that we need the exact location of all the agents, 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 lets 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 3.5 Agent Network with Stationary Frame

**

*Figure 3.5 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:

 (1)

We can replace the terms  in our weighted consensus equation with equation (1).

 (2)

The angle  is the equal to the robot ’s direction heading and the angle between that angle and the angle the robot needs to turn to face robot .

 (3)

If we substitute equation (3) into equation (2) and invoke the trigonometric identity for angle sums, we get:

 (4)

 (5)

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

 (6)

We can repeat this process for our vertical terms using the trigonometric identities for the sin(x) function:

 (7)

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 heading desired heading is not defined.

 (8)

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.

 (9)

If we substitute equations (5), (6) and (9) into (8) we get:

 (10)

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

(11)

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 11 we can see that:



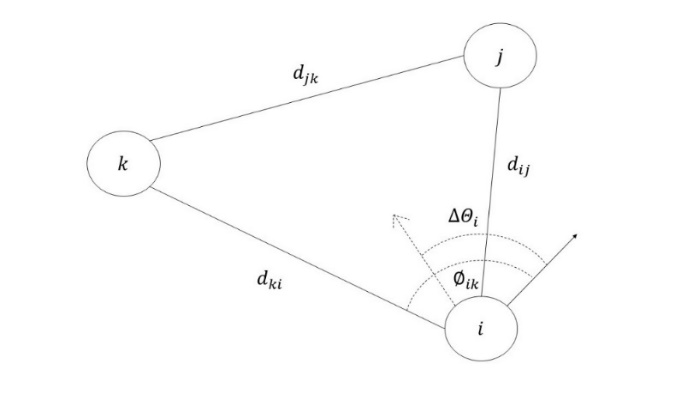


Therefore, the angle the robot will have to turn to be on the correct heading which will take it to satisfy the gradient descent from section 3.2 is:

 (12)

We can see that equation 12 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 3.6 Robot Network with Agent Frame



*Figure 3.6 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 cap the allowed speed to prevent the system from diverging.





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 [15]. 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 (12):



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. Finally, we can include a term for moving towards a target location:



This also means we need to expand our term for descried velocity which will include the relevant energies associated with robot-obstacle interactions and distance to the target location:





To avoid overwhelming the rotation inputs we can choose to shut off certain inputs depending on what state a given robot is 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 | On | Off | On | Off |
| 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 contort 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. If we have several distance sensors lining the perimeter of the robot we can determine which direction the neighbor, obstacle or target is with respect to a robot’s heading. Using this information, we can calculate , the desired robot 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, tribometry and random number generation. Finally, we used the *matplotlib.pyplot* library to obtain the data used in the figures on Total Energy. The figures that show total energy and the system behavior were taken directly from the simulation visualization.

**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:



x is a column vector containing the positions of all the agents with respect to the simulation reference frame.  is the set of agents neighboring the vertex 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, is the set of all obstacles in sensors range of robot . The second equation of motion we are testing has the form:







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 inputs , andwhich are the commanded angles based on our algorithm from section 3.6. We also use the prescribed values for  to determine the speed of each robot in the simulation.

**4.2 Test Cases:**

We submitted both control strategies to a series of tests and compared the total energy of each system with respect to the convergence and completion time. We also looked at the number of collisions with other robots and obstacles for cases where it applies.

1. *Rendezvous and achieve formation*

We will initialize random positions for the agents and allow them to converge and create a formation. We will also look at cases where the system settles in a false formation and show how both algorithms deal with these situations.

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

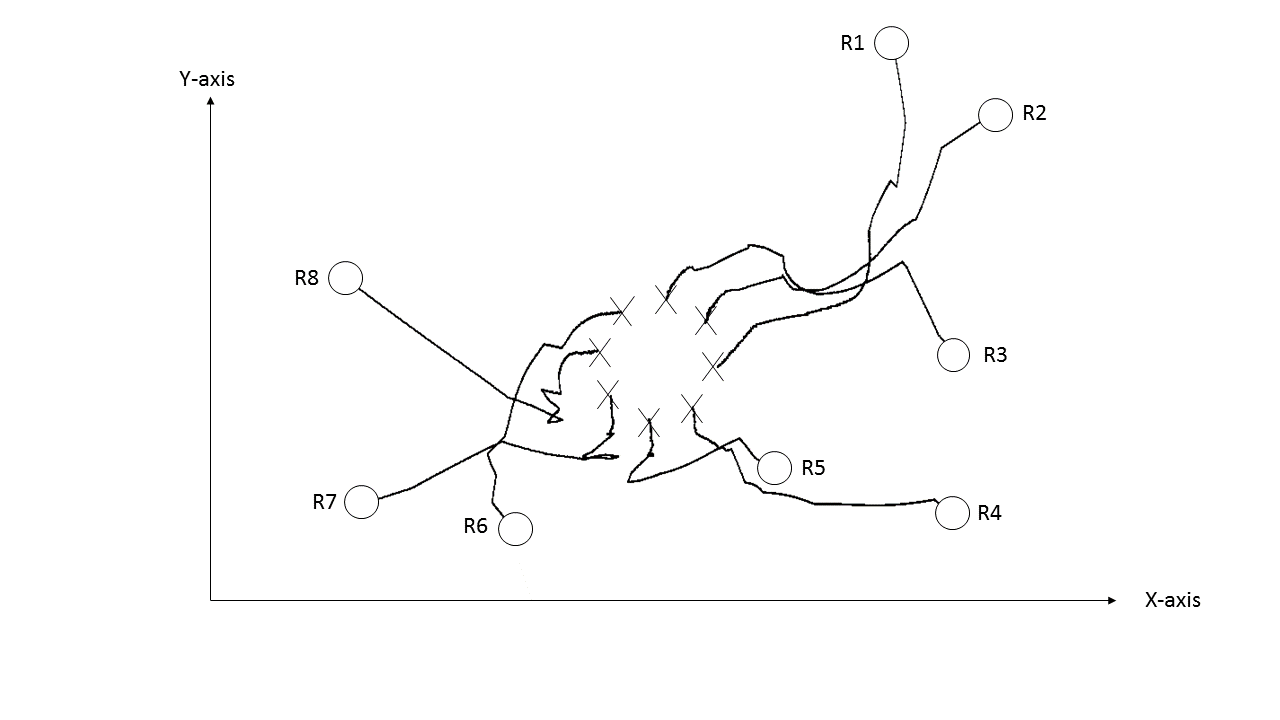
The system will be initialized already in formation and be allowed to navigate a static obstacle course.

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

The system will be initialized in formation and be allowed to navigate through a dynamic obstacle course.

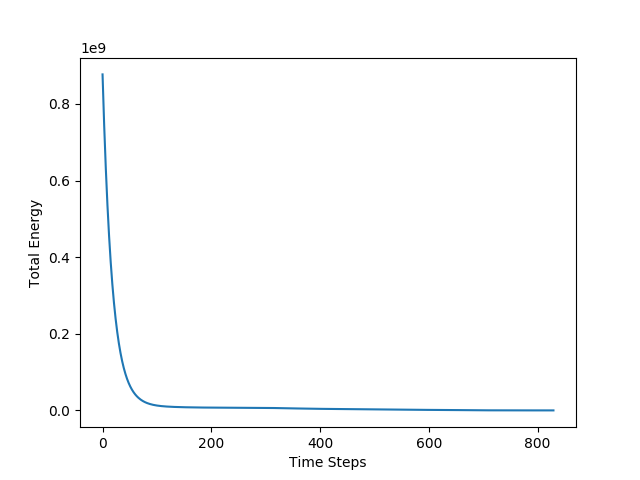
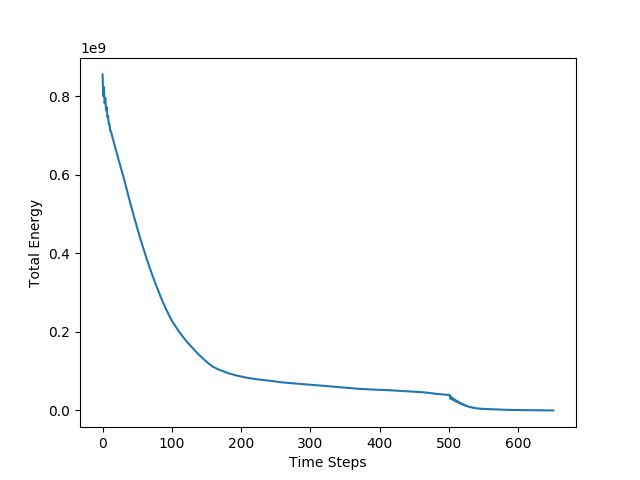
**4.3 Simulation Results:**

Figure 4.1 Convergence to Formation

****

*Figure 4.1 shows the path of all agents as they converge to a formation.*

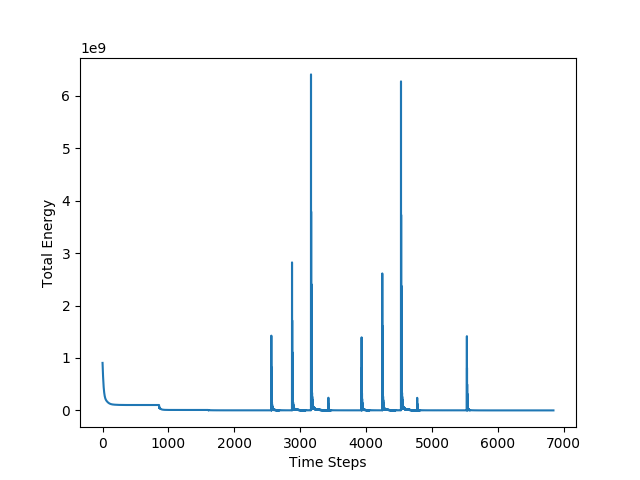
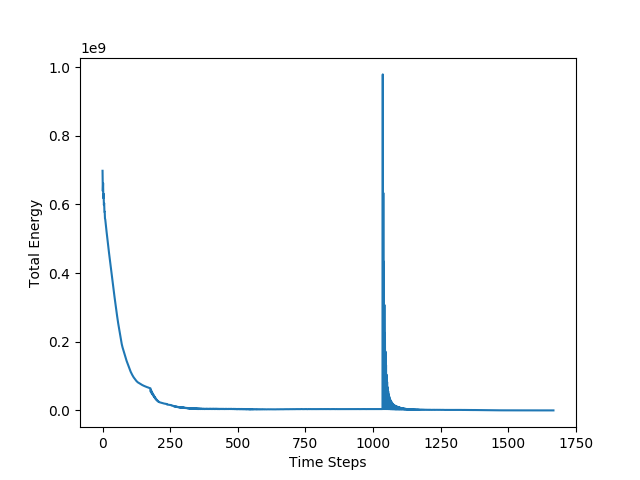
Figure 4.2 Total Energy during Formation Convergence

*Figure 4.2 shows the total energy of both algorithms as the system converges to a formation. The figure on the left the shows the energy for the absolute position based algorithm while the figure on the right shows the energy for the robot frame based algorithm.*

The agent positions are initialized randomly and allowed to move according to the control scheme developed in Chapter 3. Both the fixed frame and robot frame based algorithms exhibited behavior like that shown in the figure 4.1. The fixed frame algorithm drops to lower energy faster than the robot fixed frame and asymptotically approaches zero. The Robot Fixed Frame Algorithm approaches the lower energies more slowly but achieves formation in fewer time step. This was the typical behavior for both algorithms observed for several test runs.

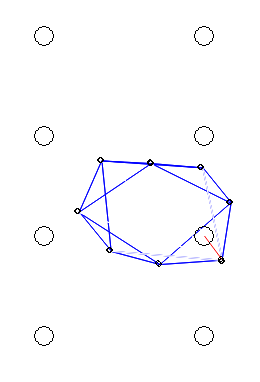
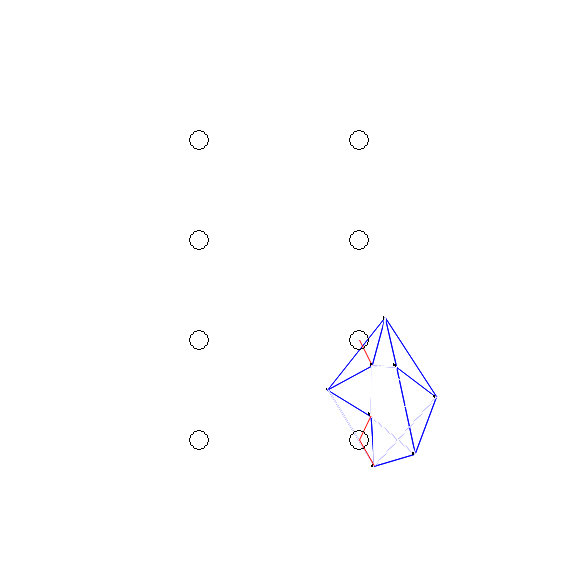
Figure 4.3 Total Energy during Convergence and Local Minima Avoidance

******

*Figure 4.2 shows the total energy of both algorithms as the system converges to a formation and avoids a local minimum in the energy. The figure on the left the shows the energy for the absolute position based algorithm while the figure on the left shows the energy for the robot frame based algorithm.*

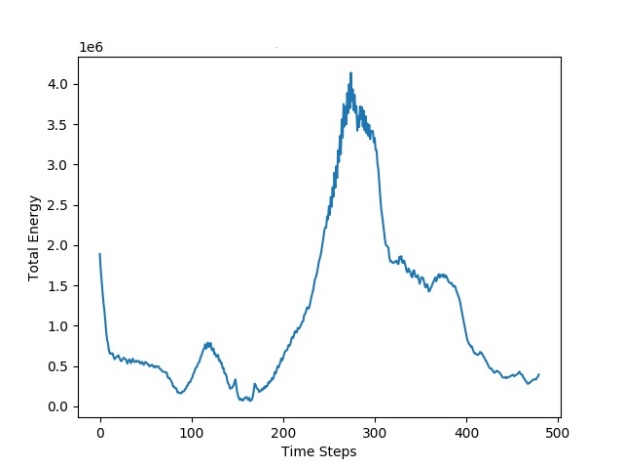
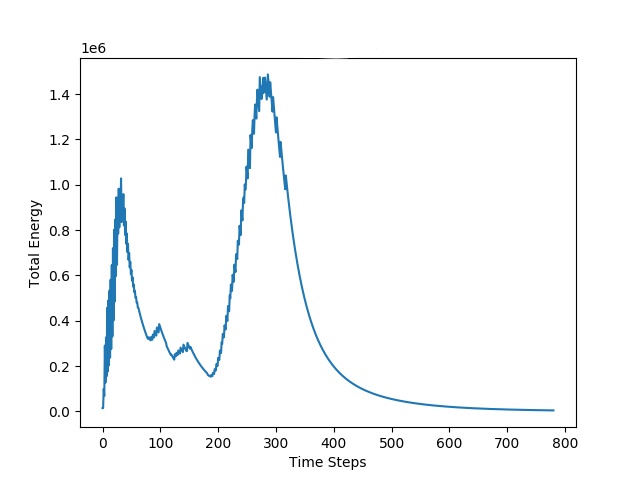
The spikes in energy are caused by the temporary connection between previously unconnected vertices in the system graph. Temporary connections in the system graph resets the convergence procedure so that it may try again. For the local minimum avoidance algorithm, we chose the following parameters:  ,and . The Absolute Position Algorithm always required several iterations through the Local Minimum Avoidance procedure to achieve the global minimum while the Robot Frame Algorithm typically only required one iteration. This is characterized by the number of spikes for each energy graph. It is possible that with a more rigorous exploration of the algorithm parameters could lead to a more finely tuned algorithm that could speed the time towards convergence.

Figure 4.4 Maneuvering through a static obstacle course.

** **

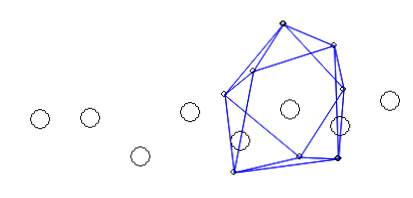
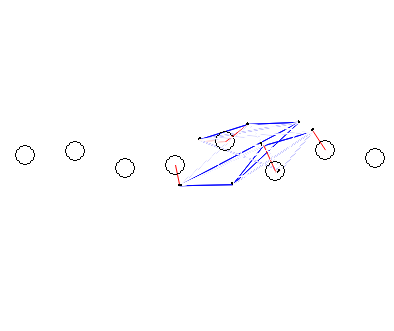
*Figure 4.4 shows both algorithms navigate through two rows of static obstacles. The figure on the left the shows the visualization for the absolute position based algorithm while the figure on the right shows the visualization for the robot frame based algorithm.*

*Figure 4.5 Total Energy during Static Obstacle Avoidance*

****

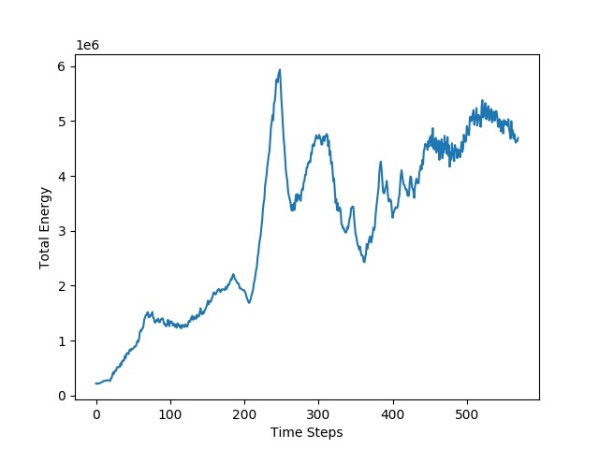
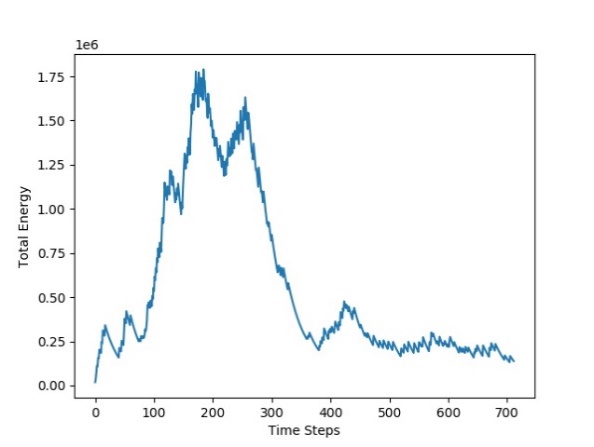
*Figure 4.3 shows the total energy of both algorithms as the system navigates through a set of static obstacles. The figure on the left shows the energy for the absolute position based algorithm while the figure on the right shows the energy for the robot frame based algorithm.*

Figure 4.6 Maneuvering through a Dynamic Obstacle Course

** ****

*Figure 4.4 shows both algorithms navigate through a row of moving obstacles. The figure on the left shows the visualization for the absolute position based algorithm while the figure on the right shows the visualization for the robot frame based algorithm*

Figure 4.7 Total Energy during Dynamic Obstacle Avoidance

**

*Figure 4.5 shows the total energy of both algorithms as the systems navigate through a set of moving obstacles. The figure on the left shows the energy for the absolute position based algorithm while the figure on the right shows the energy for the robot frame based algorithm.*

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 the in higher energy spikes in the energy graph. The Robot Frame Algorithm’s total energy jumped to maximum of 4x106 units while avoiding static obstacles. Meanwhile, the Absolute Position Algorithm’s total energy reached 4x106 units. 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 can bring it down to a lower level.

Table 3 Collision Events

|  |  |  |
| --- | --- | --- |
|  | Dynamic | Static |
| Absolute Position | 1 | 0 |
| Robot Frame |  |  |

*Table 3 shows the average number of collisions incurred by both algorithms in both dynamic and static environments.*

**Chapter 5: Conclusions**

From the data collected we saw that

**References:**

[1] Falconi, Riccardo et al. “Edge-weighted consensus-based formation control strategy with collision avoidance.” Robotica 33 (2015): 332-347.

[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. ASME. J. Dyn. Sys., Meas., Control. 2016;139(2):024501-024501-7. doi:10.1115/1.4034745.

[4] Nicholas Hoff, Robert Wood, Radhika Nagpal. Distributed Colony-Level Algorithm Switching for Robots Foraging. In Distributed Autonomous Robotic Systems 9. Berlin Heidelberg: Sringer-Verlag, No Date Available.

[5] Hartono P., Kakita S. (2008) Embedded Neural Network for Swarm Learning of Physical Robots. In: Kůrková V., Neruda R., Koutník J. (eds) Artificial Neural Networks - ICANN 2008. ICANN 2008. Lecture Notes in Computer Science, vol 5164. Springer, Berlin, Heidelberg

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

[7] R. Pedrami, S. Wijenddra, J. Baxter, B.W. Gordo. A Control Allocation Approach for Energetic Swarm Control. IEEE International Conference on Robotics and Biomimetics, 2008. ROBIO 2008.

[8] MCR. Van der Klauw. Mobile Robot Swarming using radio signal strength measurements and dead-reckoning Master’s Thesis from TUDelft Center for Systems and Control. October 2015

[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. 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, 2009. NaBIC 2009.

[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] 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.