12-9-2016

Asset Protection in Escorting using Multi-Robot Systems

Stephen Powers
sp7494@rit.edu

R · I · T

Asset Protection in Escorting using Multi-Robot Systems

by

Stephen Powers

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Science

Department of Computer Science

B. Thomas Golisano College of Computing and Information Sciences

Rochester Institute of Technology

Rochester, New York

December 9, 2016

Committee Approval:

Zachary Butler Date

Committee Chairman/Thesis Advisor

Aaron Deever Date

Committee Member/Reader

Sean Strout Date

Committee Member/Observer

Dedication

To my parents, who have encouraged me to pursue my goals no matter how difficult or outlandish. To my brothers, who certainly aren’t afraid to tell me how outlandish a goal is. And to my friends, without whom my sanity would have died long ago.

Acknowledgments

I would like to thank Dr. Zachary Butler who has consistently provided support and advice throughout this experience. I would also like to thank Dr. Aaron Deever and Professor Sean Strout for their time, patience, and support as well.

Abstract

Swarm robotics is a field dedicated to the study of the design and development of certain multi-robot systems. Often times, these groups prove to be more beneficial than a single complex robot as swarms typically provide a more robust and potentially more efficient solution. One such case is the task of escorting a specified target while addressing any potential threats discovered in the environment. In this work, a control algorithm for a high volume, decentralized, homogeneous robot swarm was developed based upon a technique commonly used to model incompressible fluids known as Smoothed Particle Hydrodynamics (SPH).

This proposed solution to the asset protection problem was tested against a more commonly accepted method for robot navigation known as potential fields. An alternate algorithm was developed based on this technique and manipulated to perform the same basic duty of asset protection. Both algorithms were tested in simulation using ARGoS as an environment and Swarmanoid's Footbots as robot models. Five experiments were run in order to examine the functionality of both of these algorithms in relation to formation control and the protection of a mobile asset from mobile threats. The results proved the proposed SPH based algorithm comparable to the potential fields based method while minimizing the escape window and having a slightly higher response rate to introduced threats. These results hint that the concept of using fluid models for control of high volume swarms should further be explored and seriously considered as a potential solution to the asset protection problem.

Table of Contents

Dedication ..... **i**

Acknowledgments ..... **ii**

Abstract ..... **iii**

Table of Contents ..... **iv**

List of Tables ..... **vi**

List of Figures ..... **vii**

List of Symbols ..... **ix**

Chapter 1: Introduction **1**

1.1 - Problem Statement 3

1.2 - Background 5

1.2.1 - Artificial Physics 5

1.2.2 - Artificial Potential Fields 7

1.2.3 - Smoothed Particle Hydrodynamics 11

1.2.4 - Communication Techniques 14

Chapter 2: Related Works **16**

2.1 - Artificial Physics 16

2.2 - Potential Field Based Asset Protection 18

2.3 - Smoothed Particle Hydrodynamics 20

2.4 - Surface Tension and Adhesion 22

2.5 - Passive Communication 23

Chapter 3: Implementation **25**

3.1 - Smoothed Particle Hydrodynamics Approach 25

3.1.1 - Communication 28

3.1.2 - Robot States 29

3.1.3 - Obstacle Avoidance 34

3.2 - Analysis Procedure 36

3.2.1 - Alternate Algorithm: The Potential Field Approach 36

3.2.2 - Experiments 39

Experiment 1 - Stable Formation Test 40

Experiment 2 - Offset Initialization Formation 41

Experiment 3 - Mobile Asset....41
Experiment 4 - Threat Pull....42
Experiment 5 - Complete Simulation....43

Chapter 4: Results....44
4.1 - Data....44
4.1.1 - Stable Formation....44
4.1.2 - Offset Initialization....47
4.1.3 - Mobile Asset....52
4.1.4 - Threat Pull....55
4.1.5 - Complete Simulation....57
4.2 - Discussion....60
4.2.1 - Stable Formation....60
4.2.2 - Offset Initialization....61
4.2.3 - Mobile Asset....63
4.2.4 - Threat Pull....64
4.2.5 - Complete Simulation....65
4.3 - Limitations....66

Chapter 5: Conclusion....69
5.1 - Summary....69
5.2 - Future Work....71

Bibliography....75

Appendix A....78
User’s Guide....78
Framework....79
Controllers....79
Arena....81
Example Configuration File....83

List of Tables

3.1 - Simplification constants.....28

3.2 - Table depicting the ordering of the ten bytes of data communicated between robots.....29

3.3 - Default parameters for all experiments.....40

List of Figures

1.1 - Potential Fields basic example....8

1.2 - Potential Force magnitudes and resulting vector fields....9

1.3 - Example of the Local Minima problem....11

1.4 - Graph of a commonly used cubic spline function for SPH with h = 1.5..12

3.1 - High level overview of the proposed algorithm's flow....27

3.2 - A common formation of 120 robot swarm around a single asset....31

3.3 - Potential Field Magnitude plot for the obstacle avoidance algorithm....35

3.4 - The force exerted on an agent by an asset when using the alternate potential fields based method....37

3.5 - Force exerted on the agent from other agents in the swarm....37

3.6 - Attractive and repulsive force exerted on an agent by a threat....39

4.1 - Example of a stable formation for a 100 robot SPH based swarm....45

4.2 - Example of a stable formation for a 100 robot potential field based swarm....45

4.3 - Scatter plot of the times for swarms of various sizes to reach a stable formation....46

4.4 - Plot of average time of swarms to reach a stable formation....47

4.5 - The starting positions of the 10, 50, and 100 robot swarms for offset initialization....48

4.6 - The formation of two 10 robot, SPH and potential field based swarms not initialized uniformly around the asset....48

4.7 - The formation of a 50 robot swarm using both SPH and potential fields and not initialized uniformly around the asset....50

4.8 - The formation of a 100 robot potential field based swarm not initialized uniformly around the asset....51

4.9 - The formation of a 100 robot SPH swarm not initialized uniformly around the asset....51

4.10 - The formation of a 100 robot Potential Field based swarm following a mobile asset....53

4.11 - The formation of a 100 robot SPH based swarm following a mobile asset.....54

4.12 - A 50 robot SPH based swarm reacting to 6 threats.....55

4.13 - A 50 robot potential field based swarm reacting to 6 threats.....56

4.14 - A graph of the number of agents reacting to variable number of threats 56

4.15 - A graph of the number of agents pulled into outer layers of the swarm in response to a variable number of threats.....57

4.16 - A complete simulation of a SPH based swarm.....58

4.17 - A complete simulation of a potential field based swarm.....59

List of Symbols

$\vec{v}_i$	Velocity vector of particle i
$\vec{g}$	Gravitational force vector
ρ_i	Density of particle i
p_i	Pressure of particle i
μ	Fluid viscosity coefficient
m_i	Mass of particle i
$\vec{x}_i$	Positional vector of particle i
h	Smoothing length
γ	Surface tension coefficient
$\vec{n}_i$	Normal vector for particle i
ρ_0	Fluid resting density
β	Adhesion coefficient
δ_{b_i}	Boundary particle density
σ	Maximum robot speed

Chapter 1

Introduction

The field of robotics is a vast area of study that spans over multiple disciplines. Whether it be mechanical engineering, electrical or computer engineering, computer science, or mathematics, all have a part to play in the field of robotics. However, the list of contributors certainly does not end with those disciplines. Robotics has benefited also from disciplines such as biology, chemistry, physics, and even the arts. With so many long established disciplines holding such strong influences in a relatively new field, there is little wonder that the field itself is so broad and may include everything from kinematics, to control systems, to material research, to mechanical design, and beyond.

Like any discipline, there are also fields within Robotic Engineering that one could spend a lifetime researching. One such subfield is that of Cooperative Robots – or multi-robot systems. To get more specific, there are different types of multi-robot systems, the two most common of which are robot teams and robot swarms. A team of robots is generally considered to be a heterogeneous group where all agents in the team have unique roles and duties to fulfill - even if they have the same hardware and system design. A swarm of robots, however, is a much more homogeneous group. A swarm is made up of many simple (and relatively inexpensive) robots that all have the same function and software. The idea stems from biology and insect colonies like those of ants

or bees. In these situations, each individual member is simple, weak, and unable to perform many duties necessary to guarantee the preservation of the species. However, despite each individual member seeming unintelligent, when placed together in a large swarm and allowed to interact with others of its kind, what's known as a 'Hive Mind' begins to reveal itself. By working together, complex behaviors necessary for survival of all individuals occur in the colony, and mere insects are able to achieve relatively amazing goals.

These biological systems inspire many control systems for swarm robotics today, and the practical uses are just starting to be discovered. Much of the research in the past couple decades has dealt with either specific tasks - such as aggregation, group formation, and environmental mapping - or more complex jobs called missions. Some of the more common missions that have thus far been studied in relation to multi-robot systems are search-and-rescue, foraging, and entrapment or escorting. This last mission is what this work will be addressing and will be discussed shortly.

One important idea vital to the control systems of any robot swarm must be addressed first, and that is the difference between a centralized and a decentralized swarm. A centralized system is one in which the main controller is a single entity and commands all agents in the group. All planning and processing is done by the central controller. The benefit to this is that the control algorithm is generally much simpler and easier to predict. All of the agents in the swarm also don't necessarily need strong processing power considering they merely follow commands. However, the major disadvantage is the scalability of this system. Controlling a handful of agents in an environment may be a simple task for the controller, but when the volume of agents

reaches into the hundreds and possibly thousands, the computational costs for calculating all necessary information may be too great for a single controller to bear. In addition to the costs, the overall swarm is dependent on one sole member. If the central controller is damaged or stops communication for any reason, the entire swarm loses all functionality.

However, with a decentralized system, each individual agent processes the local environmental information and acts by itself, potentially interacting with other members of this swarm. This distribution of control does make system planning slightly more difficult and harder to predict, though it reduces the need for a central controller. Now, like the insect colonies, the entire swarm is scalable and will not collapse when a handful of individuals fail.

1.1 Problem Statement

This work will be focusing on an extended version of the entrapment/escorting mission. In the basic mission, the goal is to completely surround a single target (also called an ‘asset’) such that the probability of the asset escaping or intruders penetrating the defenses of the swarm are minimal. This mission highly depends on a swarm’s ability to form a protective perimeter around a target and minimize the escape window (the distance between robots in the formation through which the asset can escape or a threat can enter). The main application for such a mission is security primarily in a military setting, though other applications include surveillance and automated patrolling.

However, the majority of the research in relation to the entrapment/escorting mission makes two major assumptions. The first assumption is that the swarm is always relatively small, usually consisting of only 10 to 50 robots. Some have addressed issues

of larger swarms, but most research - for sake of a lack of resources in computing power or finances - has limited the size of their swarms to under 50 agents. This leads to another assumption that these studies would have to make in that each threat to the asset would be properly neutralized or deterred by at most two or three agents in the swarm. However, these assumptions may be dangerous to make. Take for example in the near future where a high volume system of insect-sized robots are tasked with protecting a military convoy. The number of agents within the swarm must reach well into the thousands in order to sufficiently protect the asset, and it is safe to assume that one miniature robot may not be enough to neutralize any threat.

The solution is to have the local members of the swarm react to threats by aggregating in the space between the asset and threat in order to build a stronger protective barrier in that location. However, it should be noted that no area of the perimeter may be broken and the asset exposed in the case of unseen threats approaching quickly from a separate direction. This solution also needs to be able to scale indefinitely to account for larger swarms protecting larger assets and thus should be a decentralized system in which all agents use only local communication.

The purpose of this work is to solve the problem of formation control given the extended entrapment/escorting mission presented above. Addressing the issue of threat detection is beyond the scope of this research and it will be assumed that each robot is capable of identifying the asset, a threat, and a non-threat. Due to a limitation in physical resources, this work will be done solely in simulation. As such, identification of the threat level of any object in simulation will be determined by communication techniques (as will be explained while discussing the implementation of the algorithm used). The

proposed solution will be tested against an alternate algorithm based on more commonly used methods of navigation in robotics. As these solutions will be designed for high volume swarms, they will be tested on larger groups of robots than typically used in other research projects. This work will also deal only with wheeled robots; this means that the solution will only be tested in two dimensions rather than three, though extending the solution into the third dimension will be relatively trivial.

1.2 Background

Even a seemingly simple task such as congregating around a particular target can be complex when considering the fact that a group of agents must do so smoothly and reliably without collisions or failures. Adding in a more complex environment including obstacles and threats, and the task becomes much more complicated. Some background in mathematics and computer science is needed to fully understand the problem and solution presented in this work. The main topics applicable to this work will be discussed here.

1.2.1 Artificial Physics

It is a very common trend in science and technology to look to nature when a problem arises that proves difficult for humans to solve. For instance, as already discussed, the idea for a swarm of similar, simple robots stems from entomology and observations found while studying colonies of ants. Control algorithms for teams of robots can also be found in biology from studying the flocking behavior of birds. It seems as though many of the problems mankind is facing in research have already been solved by nature, and thus it is not uncommon for researchers to create solutions that directly

imitate nature itself. Artificial Physics is one of the solutions for control algorithms based on natural principles of physics.

More specifically, the Artificial Physics (AP) framework replicates molecular interactions at a basic level. Just as atoms experience repulsive and attractive forces based on distances from other atoms, so too do individual ‘particles’ of the AP framework. There is a specified distance (which we will specify as *R*) that is deemed to be the target distance between all particles of the system. If one particle is some distance *r* away from another particle such that $r > R$, both particles will experience an attractive force to each other. However, if the distance between the two is such that $r < R$, the force is repulsive. The first case leads to a reduction in *r* while the second leads to an increase. The combination of both forces results in particles coming to a rest approximately *R* units away from one another. In a complex system involving many particles, the combined force acting upon a particle may be impossible to compute considering the lack of the ability for a particle to see infinitely. This is why the particle only calculates and combines forces from other particles within a given distance (generally as far as the particle can ‘see’, or usually 1.5*R*).

The position of the particles is influenced by the velocity derived from the force equation. This equation is $\Delta v = F\frac{\Delta t}{m}$, where Δt is the time step and *m* is the virtual mass of the particle. Obviously, this does not have to be the physical mass of the robot or object in the swarm, but rather a preconceived number to allow the calculations to work out well. For simplicity’s sake, the mass of any particle is usually just assigned to

be 1. The equation to determine the force is given as follows: $F = G\frac{m_i m_j}{r^2}$ where r is the distance between the two particles, m is the masses of the particles i and j (again, simply 1 in the most general case), and G is a gravitational constant. Much like the mass, the gravitational constant is there to provide flexibility in the equation. The higher the constant, the stronger the forces and the larger the velocity of individual particles in the system [20].

Other researchers have also used this framework but substituted the Morse Potential Function in for the general force equation. The Morse Potential Function better emulates atomic attractions and allows for much more flexibility in controlling the attractive and repulsive forces individually. Once again, this formula stems from molecular physics and serves to illustrate how the overall Artificial Physics framework can be extended to allow more complex behavior in a multi-agent system [11]. Considering the use of artificial physics in this work will not include advanced concepts such as the Morse Potential, there is no need to go into greater detail here.

1.2.2 Artificial Potential Fields

One of the most useful and hence most common methods of control for robotics based on principles of physics is the use of artificial potential fields (APF). These stem from the attractive and repulsive forces in magnetic fields and at its core is very similar to the AP framework. The concept is that the agent being controlled is similar to a magnet, and all obstacles in the immediate vicinity apply some type of force (either repulsive or attractive) on the agent. An obstacle, for instance, would have a repulsive force and the

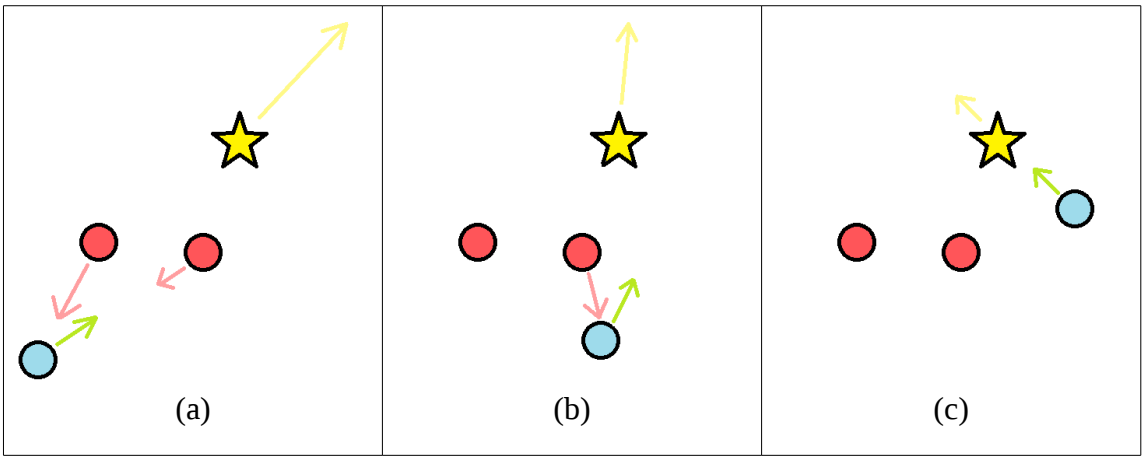
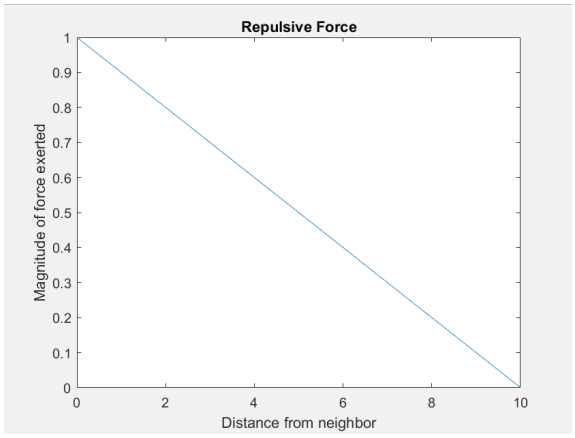


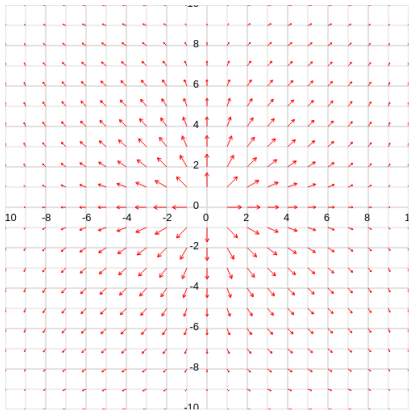
Figure 1.1. Agent (blue circle) calculates the resulting vector (green arrow) from the repulsive forces (red arrows) exerted by the obstacles (red circle) and the attractive force (yellow arrow) exerted by the goal (yellow star). The agent combines all three vectors at first (a), though once the robot is sufficiently far from an obstacle, the agent no longer considers the obstacle (b), and once all obstacles no longer impede the agent, only the vector to the goal is considered (c).

end goal location for the agent would have a strong attractive force. Each object has a vector field associated with it (where the vector would represent a velocity), and the agent would experience the combination of all vectors at its current location.

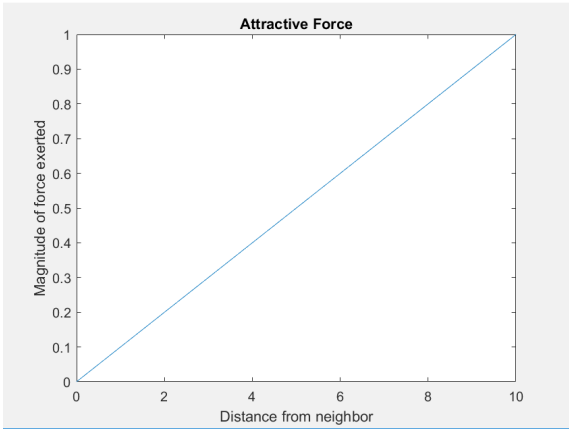
Take for instance the example presented in Figure 1.1. In part a, the agent sees two obstacles and calculates what the vectors of both of their potential fields would be at its current location. It also calculates the goal point vector and then adds all three together. This creates a single vector which represents the velocity the particle should use to get to the goal location. Figure 1.1 (b) shows the agent some time later and the resulting vectors from a single obstacle and the goal along with the combined vector as well. Only one obstacle needs to be considered, as the robot is sufficiently far from the other and it is not in between the robot and its goal. Finally, in Figure 1.1 (c), we see that the agent only needs to calculate the vector to the goal as no obstacles in the environment



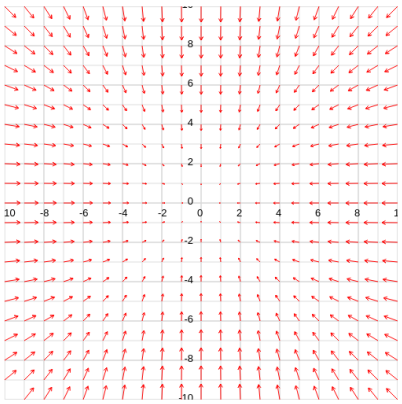
(a)



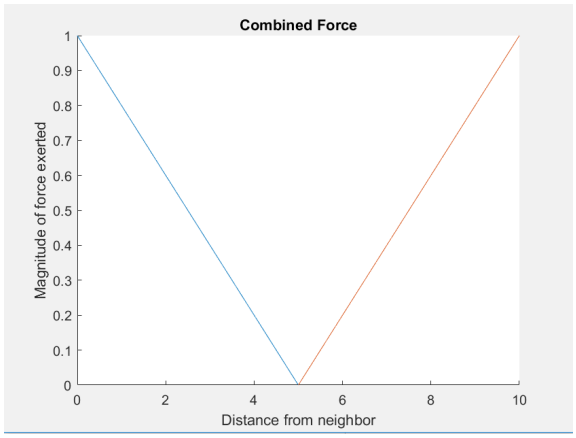
(b)



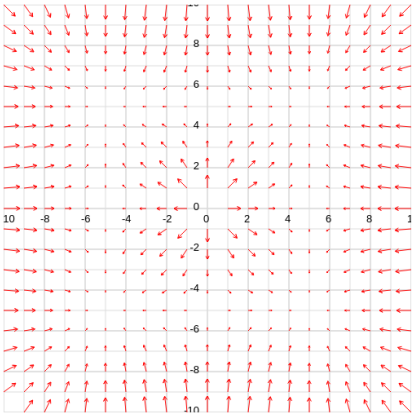
(c)



(d)



(e)



(f)

Figure 1.2. Magnitude of repulsive force (a), resulting repulsive potential field (b), magnitude of attractive force (c), resulting attractive field (d), combination of attractive and repulsive (e), along with the resulting field (f).

are obstructing its path. The beauty in this method is not only in how well it works in practical applications, but also the simplicity of the math. The magnitudes of the vectors in the field of a single obstacle can be represented by a simple distance equation such as that shown in Figure 1.2 (a). The closer the object is to the obstacle, the stronger the repulsive force, though it weakens and eventually disappears after some distance away. The directions of the vectors is also simple to calculate as they are normal to the surface of the obstacle. For instance, the vector field resulting from the equation in Figure 1.2 (a) is shown in Figure 1.2 (b). For an attractive force, the equation is the opposite such as that of Figure 1.2 (c). Generally, there is a capacity limit to the magnitude of the vectors considering a robot can only safely move so fast, but one can see that the magnitude of the force starts to decrease as the distance from the agent to the goal decreases to allow for a stable approach. The direction of all vectors are merely the inverse of the normal, resulting in a vector field such as that in Figure 1.2 (d). In addition to purely attractive and purely repulsive forces, one can also have a slightly more complex vector field by representing the magnitude as a piecewise equation such as that in Figure 1.2 (e) and switching the direction of vectors at the minimum of that equation. This results in the agent experiencing a repulsive force from an obstacle if it is close and attractive force if it is farther away, stabilizing some safe distance away from the goal as evident by the resulting vector field in Figure 1.2 (f). As one can see, this looks very similar to what the AP framework accomplishes.

Unfortunately, potential fields have one major weakness, and that is the possibility to run into what is known as the local minima problem as shown in Figure 1.3. This problem occurs when the vectors calculated from all obstacles and the goal location result in a zero vector. Since the sum of the repulsive forces of the obstacles are just as strong as the attractive force of the goal, and the directions are opposite, the resulting vector tells the agent to stay exactly where it is. The robot then becomes stuck in this single location because neither its location nor the location of all points of interest change and the robot never reaches the goal. Granted, there are many ways to avoid or solve this problem, but this must always be explicitly handled inside of the control algorithm.

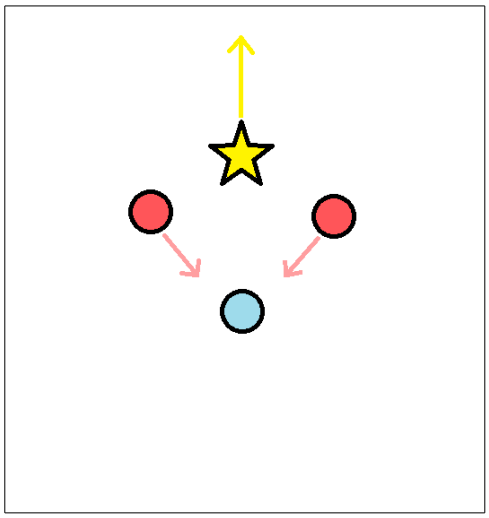


Figure 1.3. An agent (blue circle) is attempting to get to the goal (yellow star), but the vectors from the goal and the obstacles (red circles) cancel one another, leaving a net velocity zero vector.

1.2.3 Smoothed Particle Hydrodynamics

Another, much less common, method of controlling a large number of robots is through the use of fluid mechanic models. Unfortunately, many of these modeling methods treat fluids as a continuous substance and are not able to be applied to finite groups of individuals. However, there is one technique known as Smoothed Particle Hydrodynamics (SPH) which estimates fluid properties by dividing the overall fluid into individual particles. At that point, discrete mathematics can be applied to the structure

and decent estimates can be made. More specifically, SPH is a technique which allows for estimation of the incompressible Navier-Stokes equations which are used to determine natural fluid flow based on pressure, the viscosity of the fluid, and gravity. Considering the complexity of the Navier-Stokes equations and SPH in general, only the basics which relate to this particular study will be explained.

What is first needed in order to compute fluid flow is known as the material derivative: the rate of change in velocity in respect to time of a given material (in this case, a fluid). Using the Navier-Stokes equations, this can be calculated for a single particle *i* as follows:

$$\frac{d\vec{v}_i}{dt} = \vec{g} - \frac{\nabla p}{\rho_i} + \frac{\mu}{\rho_i}\nabla^2\vec{v} \qquad (1)$$

Where $\vec{g}$ is the gravitational constant, ρ_i is the density of the particle, ∇p is pressure gradient, and $\frac{\mu}{\rho_i}\nabla^2\vec{v}$ is the viscosity term governed by a viscosity constant μ and a function of the velocity.

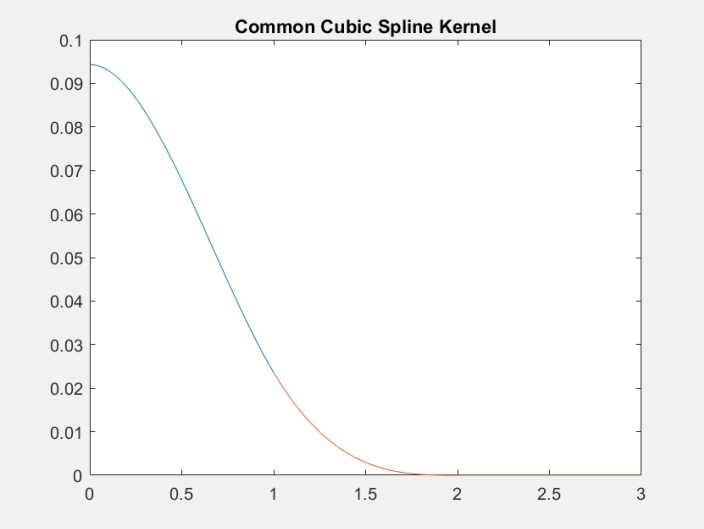


Figure 1.4. Graph of a commonly used cubic spline function for SPH with $h = 1.5$.

As previously mentioned, SPH can estimate this material derivative by separating the fluid into particles and introducing two new concepts. First, there is the idea of a smoothing kernel (generally denoted as *W* in any equations). The technique uses surrounding particles to that being evaluated in order to estimate given properties (in this case the material derivative), and the smoothing kernel establishes the weights to which each neighboring particle influences the properties. For instance, one very common type of smoothing kernel is the cubic spline function depicted in Figure 1.4. If this function is used, particles closer to that being evaluated would have a much higher influence on the outcome of the calculation relative to those farther away. After some distance, the impact of particles becomes negligible. This distance is the second concept introduced and is usually found by another parameter known as the smoothing length (or rather simply *h*).

Once these concepts have been established (and history does reveal which smoothing kernels seem to be the best for natural simulation), the three terms in the Navier-Stokes can be estimated with knowing the masses and velocities of neighboring particles. The necessary equations for proper estimation – including the smoothing kernels – which are best used for SPH simulation are as follows.

$$\rho_i \approx \sum_j m_j W(\vec{x}_i - \vec{x}_j, h) \tag{2}$$

$$\frac{\nabla p_i}{\rho_i} \approx \sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W(\vec{x}_i - \vec{x}_j, h) \tag{3}$$

$$\frac{\mu}{\rho_i} \nabla^2 \vec{v}_i \approx \frac{\mu}{\rho_i} \sum_j m_j \left(\frac{\vec{v}_j - \vec{v}_i}{\rho_j} \right) \nabla^2 W(\vec{x}_i - \vec{x}_j, h) \tag{4}$$

$$W(\vec{x}_i - \vec{x}_j, h) \equiv \frac{315}{64 \pi h^9} (h^2 - \|\vec{x}_i - \vec{x}_j\|^2)^3 \tag{5}$$

$$\nabla W(\vec{x}_i - \vec{x}_j, h) \equiv \frac{-45}{\pi h^6}(h - \|\vec{x}_i - \vec{x}_j\|)^2 \frac{\vec{x}_i - \vec{x}_j}{\|\vec{x}_i - \vec{x}_j\|} \qquad (6)$$

$$\nabla^2 W(\vec{x}_i - \vec{x}_j, h) \equiv \frac{45}{\pi h^6}(h - \|\vec{x}_i - \vec{x}_j\|) \qquad (7)$$

Through the combination of these equations SPH can accurately estimate the material derivative of any incompressible fluid. It should be noted that while these equations seem complex and computationally expensive, several parameters of the particles are determined outside of simulation, and hence many of these terms simply become constants and can be computed outside of the actual runtime.

1.2.4 Communication Techniques

Even in the biological systems from which swarm robotics draws inspiration, it is sometimes necessary for members to communicate in order to work together more efficiently, and nature presents several ways of communication between members. The most obvious happens to be through the use of voice or audible sounds such as human speech. This is referred to as active communication in which two or more members directly communicate with each other. In a robotic swarm, this can be simulated using multiple different technologies such as wireless networking or infrared devices. Here a robot can explicitly share information with a neighboring member of the swarm.

Unfortunately, such communication is generally expensive in terms of power or near impossible depending on the size of a swarm. And thus when the information that is needed to be communicated is simple, it seems like the amount of power lost in communication is not worth the information gained. In addition, communication between

the whole swarm has the potential to become unreliable or costly in terms of time as the number of agents increase as well.

But once again, nature solves the issue of a lack of active communication through passive communication techniques. One such example in nature is a colony of ants searching for food. When relaying information about a path to a food source, an ant will lay down chemical pheromones which other members of its colonies can sense. This communication is not directed at any one member of the colony, yet it efficiently relays the information to anyone and everyone that passes. This is just one of the many passive communication techniques that have been adopted into the robotics research community. Generally, however, the use of virtual pheromones for communication among robots in a swarm are preferred over chemical pheromones. In this manner, the cost of communication can be greatly reduced when simple information must be transferred to all surrounding agents.

Chapter 2

Related Work

The entrapment/escorting mission has been the focus of many studies, though the shortcomings of most of the research conducted have already been discussed in section 1.1. Nevertheless, there have been multiple studies that - while not necessarily addressing the problem presented in this work - provide valuable insight. Through the combination of different techniques already proposed by existing work in the field, an algorithm can be created which can efficiently allow for sufficient asset protection in scenarios where single agents in a multi-robot system are not able to provide protection. While many studies can provide valuable information, the following are the works which hold the most influence on this research.

2.1 Artificial Physics

The structure of the Artificial Physics framework has already been briefly discussed in section 1.2.1, though not much has been said about its uses and effectiveness. William Spears and Dianna Gordon of the Naval Research Laboratory first came up with the framework in 1999 [20]. The goal of the research was to establish a broad solution that could not only be used as a control algorithm in a multi-agent system, but has the potential for use in quantum computing as well. As already discussed, the

approach uses virtual forces as defined by the physics equations $\Delta v = F\frac{\Delta t}{m}$ and $F = G\frac{m_i m_j}{r^2}$. The experiments conducted were evaluated by the system's ability to construct a stable lattice after a given time [20].

What Spears and Gordon found was that the system was able to form a relatively stable hexagonal lattice. This structure is formed by default due to the geometry of the circle; all points attempt to push away from each other until they are all some distance *R* away. However, while the structure itself forms a decent hexagonal lattice, the global shape of the resulting mass is not necessarily guaranteed to be a perfect hexagon. In addition to this imperfect shape, another outcome discovered as a result of the framework is the formation of agent clusters. A final mass of agents was formed 1000 timesteps after releasing two hundred tightly clustered particles acting under the AP framework and show multiple clusters of two to three agents very close together. These clusters result from the fact that sometimes the repulsive forces of a single agent upon another is not enough to counteract the combined repulsive force of surrounding agents. Hence, one can generally find these clusters averaging three agents in size and located closer to the center of the lattice. Spears and Gordon rationalized that this clustering is not necessarily a flaw, however, as it can add in redundancy to the system and provide back-up agents in the case of failing individuals [20].

Spears and Gordon then proposed a solution to both the clustering and global formation problem, though such a solution would require a small amount of global information. If each particle received data which could be ordered, such as a pair of numbers, and each particle knew which direction in the global environment represented

an increase in the data and which represented a decrease, the system can become self-organizing and result in a perfect lattice. As long as some sorting is predetermined, this method can work for types of data other than numbers as well. Nevertheless, the same algorithm is used with a minor alteration such that two particles are attractive as well if their ordering is incorrect. This attractiveness will cause the two particles to pass each other and swap places, eventually resulting in an ordered lattice. The result of this algorithm on 217 particles is a perfect global hexagonal lattice. This proves that this method can work both in computing techniques and control algorithms for large volumes [20].

2.2 Potential Field Based Asset Protection

In 2010, Dieter Laskowski attempted to develop an algorithm to allow a considerably small swarm of robots to protect an asset from known threats [13]. His solution is based on artificial potential fields and varying magnitudes of attractive forces from assets and threats. The assets in simulation move about in random directions while the threats' intelligence is also based on potential fields. Each threat is attracted by an asset but repulsed by agents of the swarm. This behavior is rather simplistic, though the algorithm being developed is that which controls the individual agents (or 'nodes' according to Laskowski) of the swarms [13].

The potential field force exerted on nodes by an asset has a global minimum at 1.0 units away, starting with a magnitude of 4 at 0 units away, and gradually increasing back to 4 between 1.0 and 3.0 units away. It is important to notice that what is represented in this data is the magnitude of the force and not direction. In reality, this potential field

replicates the behavior of the artificial physics, such that if the node is too close to the asset (in this case, $r < 1$ unit), it will have a repulsive force. If it is too far away from the asset ($r > 1$ unit), it will have an attractive force. Much like the AP framework, this results in the nodes all coming to rest 1 unit away from assets. Similarly, the forces exerted on the nodes by the threats is repulsive - though much smaller - when the distance between the two is less than 1, and attractive when greater than 1. It is assumed that the node is able to neutralize a threat most efficiently at this distance. In relation to the potential force on nodes by other nodes, there is no attractive force needed and thus only a repulsive force is used to prevent collisions [13].

The behavior achieved from these potential fields upon first testing was good, though there were a few flaws found. The first problem was that nodes often fell into the local minima problem common when using potential fields. The second problem was when there was a greater threat on one side of the asset, all nodes would flock to the space between an asset and large threat, possibly leaving the opposite side of the asset vulnerable. This is a major flaw in the design as any threat with human-like intelligence would be able to pick up on this vulnerability and easily exploit it. The solution Laskowski presents to the first problem is rather simple - only allow the nodes to focus on a single asset to protect for the duration of the simulation. The solution for the attraction to the greatest threat, however is a little more complex. The algorithm is slightly changed such that the potential fields are replaced by Hooke's spring law. This allows nodes to keep a relatively constant distance between them to adequately surround the asset [13].

In the end, the simulation results were as expected. In the scenario of a mobile asset, when three or less nodes were used for protection, even with one threat, there was a 0 percent success rate. However, when six nodes were used to protect an asset, the algorithm achieved success more than 80 percent of the time with less than 16 threats [13].

2.3 Smoothed Particle Hydrodynamics

Smoothed Particle Hydrodynamics (SPH) has been applied to robotic swarm control by Pimenta et al. in 2013 [18]. The goal of this research was mostly to establish an effective control algorithm for decentralized, multi-agent pattern formation. In all experiments performed by Pimenta et al., a swarm of varying sizes was sent through an environment to a planned destination. Once all agents got to the goal location, they would have to create a particular pattern such as a ring or any arbitrary path. The solution was to split the control algorithm into two steps working by different methods [18].

The first step was to create a global potential function whose main job was to drive the robots from their current location to the end goal. The researchers used harmonic functions to accomplish this task considering the lack of local minima. Due to this absence, robots will not be stuck in the middle of the path and will theoretically always continue towards the goal. Considering this step is specific towards a navigation task as opposed to escorting and entrapment, the intricacies will not be discussed [18].

However, the second step is the formation of the pattern once the swarm has reached its final position. This task is accomplished through SPH with 2-D cubic splines as a kernel function and a smoothing length of 1 meter. Several experiments were run in

both simulation and on physical robots. What was found through testing in simulation was similar to the clustering problem discovered in Spears and Gordon's research involving Artificial Physics. While the harmonic function of Pimenta's first step is enough to guarantee obstacle avoidance and get a single agent from start to finish, the forces of other agents in the system tend to have unpredictable consequences. Like the clustering that formed in the hexagonal lattice, there is no more guarantee of obstacle avoidance due to inter-system forces overpowering the repulsive forces of the obstacle [18].

The solution to this obstacle avoidance problem is relatively simple. Due to the setup of SPH, all one has to do is pretend the object found is a particle in the 'fluid' being simulated. Granted, this is a virtual particle and will not move, but when Pimenta et al. placed a temporary particle at the location of the object, avoidance became much easier. Even better results were obtained through splitting the obstacle up and acting as if the object was merely a combination of similar particles in that particular shape [18].

Experiments in simulation show astounding success. Even with mobile obstacles that aren't accounted for in the global harmonic function, the swarm is able to smoothly avoid collisions and completely fill the specified shape. Unfortunately, reality proved to be a little more error prone due to uncontrollable factors such as sensor noise. However, the robots were successfully able to avoid collisions and fill the correct pattern with some slight error. This research has proven the functionality of SPH in multi-agent system control [18].

2.4 Surface Tension and Adhesion

While SPH has been around for decades and used in several fields such as astrophysics, engineering, and computer graphics, improvements are continuously being made. For instance, in 2013, Akinci, Akinci, and Teschner performed work to enhance the SPH technique in computer graphics to involve a more life-like surface area minimization and adhesion factor [2]. Standard SPH methods often result in images that seem slightly unrealistic, showing a few obvious flaws such as a choppy or non-spherical droplet of water. Using Akinci et. al.'s enhancement, however, results in a much more life-like image of fluids by minimizing surface area. Likewise, an adhesion force was added as well, which allows for more flexible interaction with non-fluid obstacles [2].

The method itself is relatively simple, following a series of equations. The first is the formula for cohesion, defined as in equation (8) below where C is a custom spline function defined in equation (9) and γ is the surface tension coefficient.

$$\overrightarrow{F_{i \leftarrow j}^{cohesion}} = -\gamma m_i m_j C(\|\vec{x}_i - \vec{x}_j\|) \frac{\vec{x}_i - \vec{x}_j}{\|\vec{x}_i - \vec{x}_j\|} \tag{8}$$

$$C(r) = \frac{32}{\pi h^9} \begin{cases} (h-r)^3 r^3 & 2r > h \wedge r \le h \\ 2(h-r)^3 r^3 - \frac{h^6}{64} & r > 0 \wedge 2r \le h \\ 0 & otherwise \end{cases} \tag{9}$$

Those two equations, though, are not enough to express the full cohesive force. There also needs to exist a surface minimization term, one dubbed as the curvature force which is dependent on the normal vectors computed by equation (10) of the neighboring particles as outlined by equation (11). This force is combined with the cohesion force as defined by equation (12) [2].

$$\vec{n}_i = h \sum_j \frac{m_j}{\rho_j} \nabla W(\|\vec{x}_i - \vec{x}_j\|) \tag{10}$$

$$\overrightarrow{F_{i \leftarrow j}^{curvature}} = -\gamma m_i (\vec{n}_i - \vec{n}_j) \tag{11}$$

$$\overrightarrow{F_{i \leftarrow j}^{st}} = \frac{2\rho_0}{\rho_i + \rho_j} \left(\overrightarrow{F_{i \leftarrow j}^{cohesion}} + \overrightarrow{F_{i \leftarrow j}^{curvature}} \right) \tag{12}$$

Finally there is the adhesion force which looks similar to the cohesion formula. This adhesion equation (13) utilizes another custom spline function A as defined by equation (14). In the adhesion equation, β is the adhesion coefficient and δ_{b_k} is the boundary particle density. Adding these forces to each particle in the fluid simulation results in a much better approximation of real life fluids and more customizable interaction between foreign objects of different materials [2].

$$\overrightarrow{F_{i \leftarrow k}^{adhesion}} = -\beta m_i \frac{\rho_0}{\delta_{b_k}} A(\|\vec{x}_i - \vec{x}_k\|) \frac{\vec{x}_i - \vec{x}_k}{\|\vec{x}_i - \vec{x}_k\|} \tag{13}$$

$$A(r) = \frac{0.007}{h^{3.25}} \begin{cases} \sqrt[4]{\frac{-4r^2}{h} + 6r - 2h} & 2r > h \wedge r \leq h \\ 0 & otherwise \end{cases} \tag{14}$$

2.5 Swarmanoid and ARGoS

The last major influence in this work is the study depicting the design and creation of a heterogeneous robot team. Dorigo et al. developed a general use heterogeneous team known as Swarmanoid consisting of three types of robots: the eyebot, handbot, and footbot [5]. All three types of robots are individuals, though they each have the ability to bond and form a physical connection with other bots in a given swarm to extend their capabilities. Footbots, whose sole objective is for navigation and

transportation, can combine with a handbot to extend its capability and allow it to not only grasp objects, but potentially climb as well. The eyebot is a flying agent that can attach itself to a metal ceiling and provide an aerial view of the swarm's environment. Different functionality and roles allows for customization of a team through varying members of individuals [5].

The goal of these robots is to be useful in both education and research. As such - since these robots were not custom made for one specific task - the creators wanted to put as much variety of hardware as possible into each robot to allow for variety in options while still having compatible technology among all agents in the team. Perhaps some of the most interesting hardware on each of the agents happens to be related to communication. While each bot does have a communication module that uses a combination of infrared and radio communication, there is one more useful module added to allow for passive communication instead. Each robot, no matter the type, possesses a ring of red-green-blue (RGB) LEDs. Considering each robot also has an on-board omni-directional camera, robots can also communicate with others in the swarm by displaying a predetermined color pattern that holds a specific meaning for the team. In this manner, as already discussed in section 1.2.4, the cost of communication among members of the swarm can be greatly reduced [5].

The creators of these robots have also developed a simulation environment known as ARGoS. This environment allows one to simulate eyebots, handbots, or footbots in a virtual space, allowing for testing of different control algorithms. Every sensor found on the real robot can be simulated in the environment, allowing for realistic behaviors and simple transportation of code from virtual simulation to reality [5].

Chapter 3

Implementation

3.1 Smoothed Particle Hydrodynamics Approach

The proposed solution to the problem depicted in section 1.1 is based mostly upon the Smooth Particle Hydrodynamics (SPH) technique commonly used in modeling fluids. As previously mentioned, SPH models a continuous fluid using discrete particles, and while its intended use was for astrophysics and common uses include computer graphics and fluid mechanics, it has shown to have some success in formation control for a decentralized swarm. As such, in situations involving asset protection in which a swarm of robots should provide a protective covering around an asset evenly until a threat is observed, SPH seems to be a useful technique. Each robot is seen as a single particle in the fluid, interacting with its neighbors based on their respective properties. However, while SPH provides the base of the control algorithm, relying solely on this technique introduces some problems including potential robot collisions among other minor issues.

Keeping with the theme of fluids, a single robot is allowed to be within one of three given states: a gaseous state, a liquid state, or a solid state. Each different state has its own purpose and its unique procedure for determining a robot's velocity. A gaseous state robot is one in which the agent is completely alone or sufficiently far from any other robots in the swarm. Any robot in this state has the goal of finding and approaching either

an asset or another agent. This differs from a liquid state robot in that a liquid state agent is already a recognized member of the swarm with the main goal of providing a barrier of protection around the asset and addressing a threat if one is detected. A solid state robot has a similar goal in that it merely wants to protect the asset, though it is not drawn to any potential threats. The solid state robots are not to be moved from their relative position around the asset for any reason other than obstacle avoidance. The secondary goal of the solid state robot is to also provide sufficient data for the innermost liquid state robots to determine their velocities as will be discussed shortly.

The high level overview of the algorithm is shown in Figure 3.1. Upon startup, all variables are initialized to predetermined values and the robot enters the main loop. The sensors are then read and a list of neighbors - along with any necessary data about the neighbors - is stored for future use. Based upon the neighbors that the robot has sensed, the state is determined and transmitted to all other robots in the swarm within range along with other necessary data such as velocity or particle density (which is used for SPH calculation). Depending upon the state, the robot calculates the velocity that it should set. This process can be done through chasing, random walking, SPH calculation, or angle locking as will be discussed in section 3.1.2. After determining the proper velocity, the robot determines if emergency obstacle avoidance is necessary and if it is, overrides the velocity to avoid collisions. After this step, the robot then sets the speed of each wheel properly and then starts back at the beginning of the main loop, performing ten iterations every second.

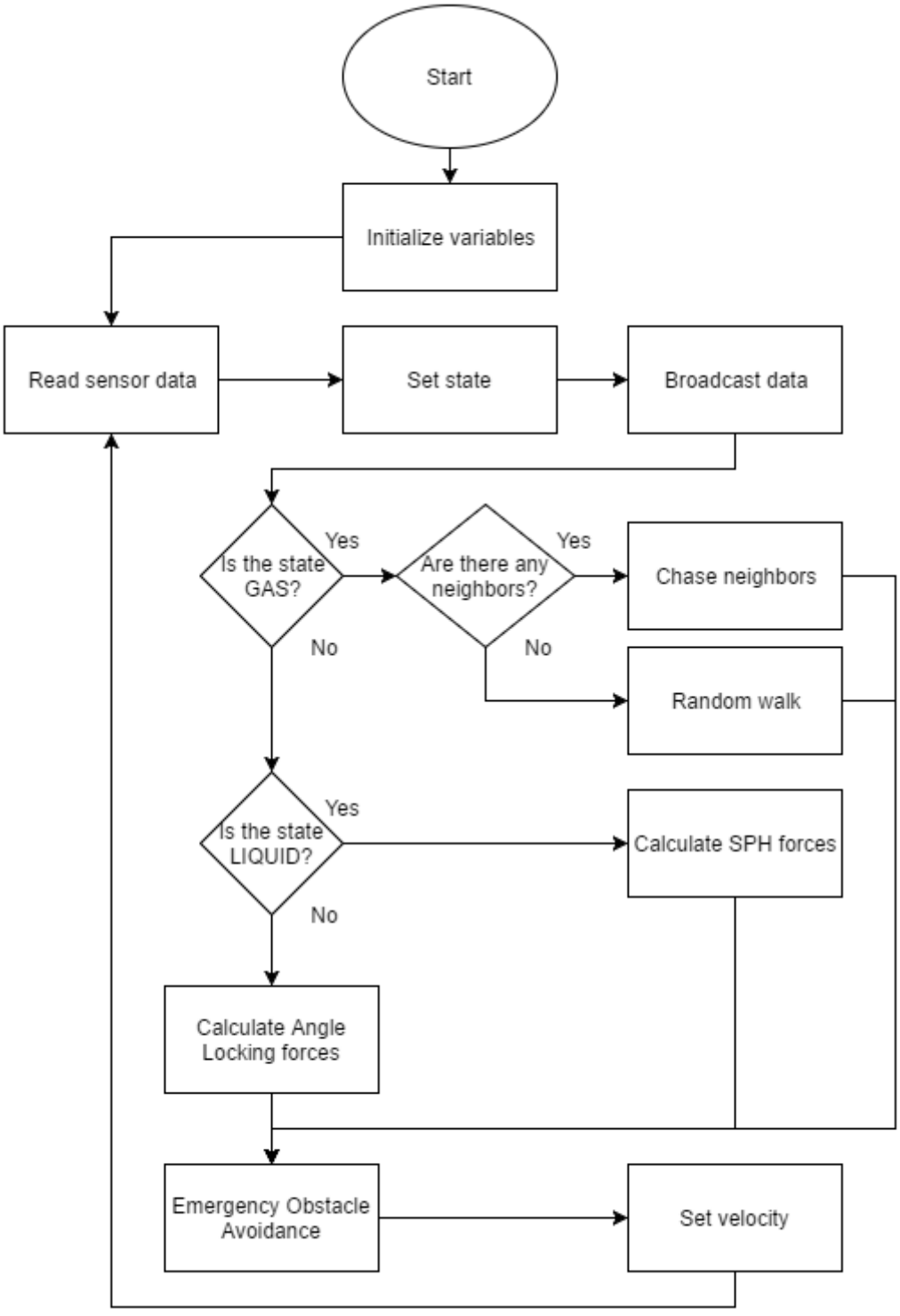


Figure 3.1. high level overview of the proposed algorithm's flow.

3.1.1 Communication

The communication abilities of each robot are very limited. Passive communication is available through the use of colored light emitting diodes (LEDs) on a ring completely surrounding each robot. Other agents can then identify the colors through omni-directional cameras. However, this proves a problem when attempting to transfer more complex data. Such a process may be useful to broadcast a robot's state, but information such as particle density, velocity, and the normal vector requires a more advanced technique for communication.

Thankfully, each footbot used also has an infrared communication device built in that allows for the ability to broadcast up to ten bytes of data for each iteration of the main loop. Unfortunately, the data that's required for the SPH calculation requires more than ten bytes of data. The particle density of the robot is expressed as a float. Then the velocity vector and the normal vector are each comprised of two floats since the work is being done in two dimensions. Since each float requires four bytes, overall, twenty bytes are needed to provide the necessary data. Thankfully, the system does not need the full precision of a float, and thus the twenty bytes can be adequately represented as ten through the use of simplification constants.

Parameter	**Value**
Velocity Vector Simplification Constant	3276.7
Particle Density Simplification Constant	25.5
Normal Vector Simplification Constant	327.67

Table 3.1. Simplification constants.

The maximum velocity that any robot can have is 10 centimeters per second, thus, mapping each signed float value in the velocity vector to two bytes is possible by multiplying the number by the maximum value of a signed short divided by the maximum velocity. Conversion back into a float is then possible by dividing by this number. Some of the precision of the original value is lost, but enough is maintained to create a stable system. This technique is used on the other data fields as well, obtaining the constants shown in Table 3.1. It should be noted that the particle density is converted into a single byte to allow room to transfer the robot's state as well (represented by a single byte). The order of transference is then represented in Table 3.2. Because this information is transferred via an infrared communication device, the robot can also detect the neighbors distance and the relative angle. Hence each neighbor's coordinates can be calculated instead of communicated.

Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
State	Particle density	Velocity Vector (X Term)		Velocity Vector (Y Term)		Normal Vector (X Term)		Normal Vector (Y Term)	

Table 3.2. Outline of the 10 bytes each robot attempts to communicate to its neighbors.

3.1.2 Robot States

As previously mentioned, a robot can be in one of three states at any given time depending on the neighbors that the robot can sense. By default, each robot is initialized to a gaseous state, though it may also enter this state if the robot is sufficiently far from every other member of the swarm or the asset. This sufficiently far distance is able to be changed, but it is recommended that this distance be less than the maximum distance at

which another agent is considered to be a neighbor. The goal of a robot in this state is to rejoin the swarm and it can do that through one of two ways: random walking or chasing.

Random walking is only used when the robot does not sense any other liquid or solid state swarm robots or the asset nearby. If this is the case, the robot moves in a random direction via a random number generator (RNG) for a random period of time between 1 and 5 seconds. As soon as it senses the asset or another robot in a liquid or solid state, the robot switches methods to chasing, where it merely chases that neighbor until it is sufficiently close and switches states entirely.

A robot will only ever become a solid state if it is within a predetermined distance of the asset (recommended value is about 0.5 meters). If this becomes the case, the robot will lock onto its current position relative to the asset. Since the robot knows the relative angle to the asset and the distance that it should remain from the asset, as long as it is fast enough, it can calculate the relative position that it should be from the asset at all times.

This method performs two essential duties. The first is that the solid state robots provide a failsafe barrier around the asset. Even if something happens to the rest of the swarm, the solid robots focus solely on the asset to determine their velocity, thus there is nothing that any outside threat can do short of destroying the agent to break the innermost barrier. The second duty these robots perform is to provide extra necessary data to the surrounding liquid state robots. As mentioned in section 3.1.1, in order to calculate the velocity using SPH, each robot needs to know its neighbor's normal vector, particle density, and velocity vector. Since it is expected that in practice, the asset will be a human or some object that does not regularly broadcast this information to the swarm, calculation of the velocity of the innermost particles would be difficult. By having this

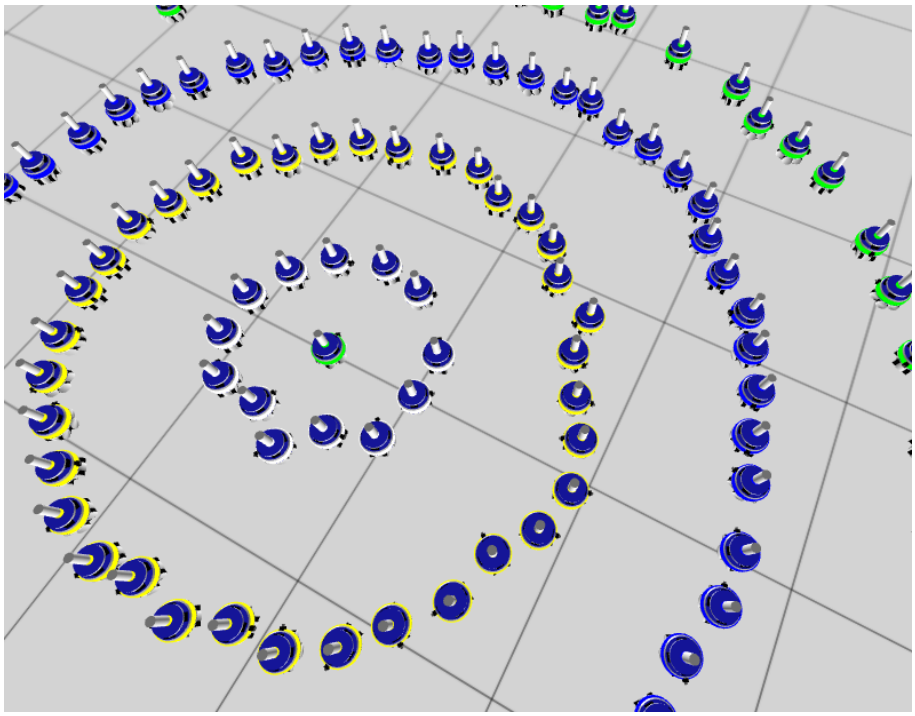


Figure 3.2. A common formation of 120 robot swarm around a single asset. The green robot in the center is the asset, white robots make up the solid state, and the outer yellow, blue, and green robots make up three layers of the liquid state.

solid state barrier that does not use SPH to determine its velocity yet can still calculate and transmit the necessary information, the problem of unknown important variables of the innermost particles is averted.

While both the solid and the gaseous states are important, the majority of the members of the swarm should be in the liquid state. If an agent is within 0.5 units of distance from another liquid or solid state member of the swarm, but not within 0.5 units from the asset, the robot will be considered to be in a liquid state. However, the term 'liquid state' may be slightly misleading as it implies that this is a single state of existence. In reality, though, there can be many different sub-states in the liquid state. Figure 3.2 shows an example of a swarm of robots in formation around an asset. The

innermost layer of robots that have the white ring of LEDs are the solid state robots. The next ring of robots that are colored yellow are liquid state robots. The blue robots are also liquid state, as they also use SPH to determine velocity, but they are in essence a different liquid than the inner ring. Likewise, the green robots on the outermost layer are also liquid state, though they're considered a different liquid than the blue and yellow members. This approach uses layered liquids and results in a target like formation which provides simplification of some calculations and enhanced performance.

When calculating the main forces stemming from standard SPH equations, each robot only focuses on neighbors apart of inner liquids. In the case of Figure 3.2, the blue robots would focus on yellow and white robots that it could see, while the yellow robots would only focus on the white. This creates a much more asset-central system, allowing each layer to focus on the attraction and repulsion from the neighbors that are closest to the asset. Thus, when the asset becomes mobile, there's no possibility for outer layers - or even the same layer - to attract the robot and prevent it from properly following the asset.

The first task of calculating the main SPH forces is to calculate the particle density of the robot itself (ρ_i). This is done by performing equation (2) for each inner layer neighbor *j* where W is the smoothing kernel which is defined in equation (5). Once the particle density is found, the pressure gradient can be calculated by equation (3) where ∇W is defined in equation (6). Like the density calculation, only particles closer to the asset are considered.

Finally, the viscosity term is calculated by equation (4), once again referring to the smoothing kernel $\nabla^2 W$ defined in equation (7). The pressure gradient and the viscosity term are then combined along with the gravitational vector by equation (1).

However, considering the calculations are done in two dimensions and gravity isn't considered, the gravitational term is merely a zero vector.

The result of the above equations is the velocity of the particle calculated by the standard SPH technique. However, based on Akinci et al.'s work [2], there can be some improvements made. For instance, using this technique alone does not necessarily fully reduce the surface area of a small volume of liquid. In order to better simulate the behavior of a real fluid, a few more forces must be calculated as well, namely the cohesion and the adhesion forces. The cohesion forces in this work will guarantee a compact swarm, drawing all agents closer to each other and closer to the asset to provide a tighter barrier. The adhesion forces will draw select agents closer to any threats that are detected in order to neutralize the threat quickly.

In accordance with the work done by Akinci et. al. [2], two different forces make up the inter-liquid attraction. For the purposes of this work, the equation used is a slightly modified version of Akinci et. al's formula shown in equation (8). By factoring in each neighboring particle's velocity along with their position, each is better able to predict the future positions of the neighboring particles and react faster, thus reducing some of the problems that occur when the entire swarm becomes mobile. The new formula is given below and makes use of the same cohesion spline function defined in equation (9).

$$\overrightarrow{F_{i\leftarrow j}^{cohesion}} = -\gamma m_i m_j C\left(\|\vec{x}_i - \vec{x}_j\|\right)\left(\frac{\overrightarrow{xv_{i\leftarrow j}}}{\|\overrightarrow{xv_{i\leftarrow j}}\|}\right) \tag{15}$$

$$\overrightarrow{xv_{i\leftarrow j}} = \frac{\vec{x}_i - \vec{x}_j}{\|\vec{x}_i - \vec{x}_j\|} + \frac{\vec{v}_j}{2\sigma} \tag{16}$$

In addition to the cohesive forces, however, one must also take into account the surface minimization forces of all neighbors closer to the asset. The equation used does not vary from that defined in section 2.4, though the combination formula does. The surface minimization term is multiplied by a scaling factor before adding it to the cohesive term. This allows for more control over the strength of the surface tension.

$$\overrightarrow{F_{i \leftarrow j}^{st}} = F_{i \leftarrow j}^{cohesive} + \psi \overrightarrow{F_{i \leftarrow j}^{curvature}} \quad (17)$$

The final force to calculate for the SPH method is the adhesion force. Thankfully, this equation may remain mostly the same as that defined in section 2.4. The only difference is that the density is not needed and can be replaced with a constant that defines the threat. Once all forces are determined, they are combined and the result is capped at the maximum speed. However, this is not yet the final velocity used by the robot.

3.1.3 Obstacle Avoidance

There is a major problem that presents itself when only considering the forces determined by SPH calculations. When this technique is used to simulate fluid, it is expected that the particles will move close to each other and have the potential to collide as that is what what happens within physical fluids as well. However, in the field of robotics, collisions between agents are generally harmful to the agents themselves. Using a technique like SPH in which a robot's velocity is determined by surrounding members means that given a large enough group, the inter-particle forces can force an agent into another robot, obstacle, or even the asset itself. This is why an emergency obstacle avoidance method possessing the potential to override the velocity previously calculated

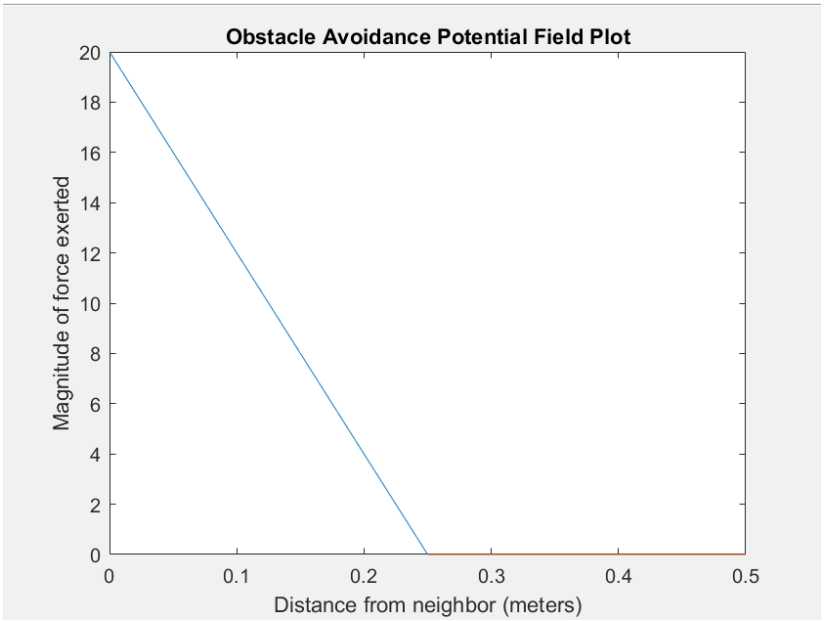


Figure 3.3. Potential Field Magnitude plot for the obstacle avoidance algorithm.

is necessary. Depending on the method used, however, there is still slight possibility of collision, but at the very least, the speed of the robot has been diminished substantially, resulting in minimal to no damage to the agent.

The obstacle avoidance method used in the proposed algorithm is the use of potential fields. At a sufficient distance away from all obstacles, this potential field force is zero, but if the robot ends up approaching a neighbor or obstacle, the robot experiences a virtual force (with varying intensity based upon distance from the obstacle or neighbor) pointing directly away and a significantly smaller force at a ninety degree angle to reduce the likelihood of a local minima. The main force used for this obstacle avoidance algorithm is plotted in Figure 3.3. Once this final force is determined and added to the velocity previously calculated in the main loop, the robot sets the speed of each of its wheels to match the intended direction and speed.

3.2 Analysis Procedure

3.2.1 Alternate Algorithm: The Potential Field Approach

In order to examine the validity of the proposed approach, a comparative analysis has been done between the SPH based control algorithm and a more common potential fields approach. This secondary approach uses only potential fields as a control method for the swarm, using the summations of attractive and repulsive forces of all neighbors. It should be noted that the use of potential fields in the most basic form to organize a protective barrier around the asset results in complete failure in high volume swarms.

When the number of agents within the swarm is small – roughly around ten agents or less – the agents spread out into a single layer surrounding the asset and generally keep this form even when the asset becomes mobile. Unfortunately, these results change drastically when the number of agents increase. As the size of the swarm increases, the number of layers around the asset increases as well. Each robot in the swarm then has a larger number of neighbors to consider when determining its own velocity. In the end, the forces from all other agents in the swarm overpowers the attractive force of the asset. Thus, when the asset becomes mobile, the robots in the swarm do not follow.

As using the potential field method in its most fundamental form will not remotely solve the problem of escorting an asset with a high volume swarm, another algorithm must be designed. This algorithm will be based upon the concept of potential fields and merely use different functions depending on the type of neighbor sensed in order to determine the virtual forces on the agent. In the case of the comparative analysis, the behavior should match that of the SPH approach as much as possible. All agents in

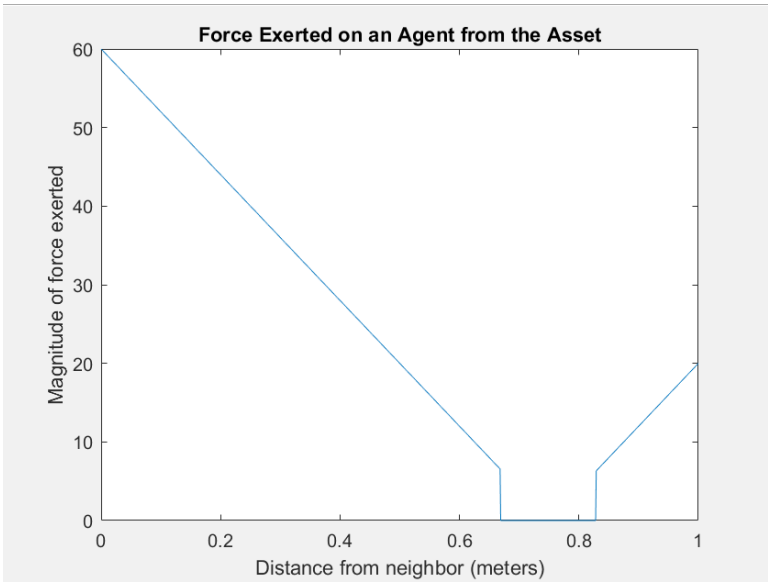


Figure 3.4. The force exerted on an agent by an asset when using the alternate potential fields based method. Ideal distance = 0.75 meters and error is 0.08 meters.

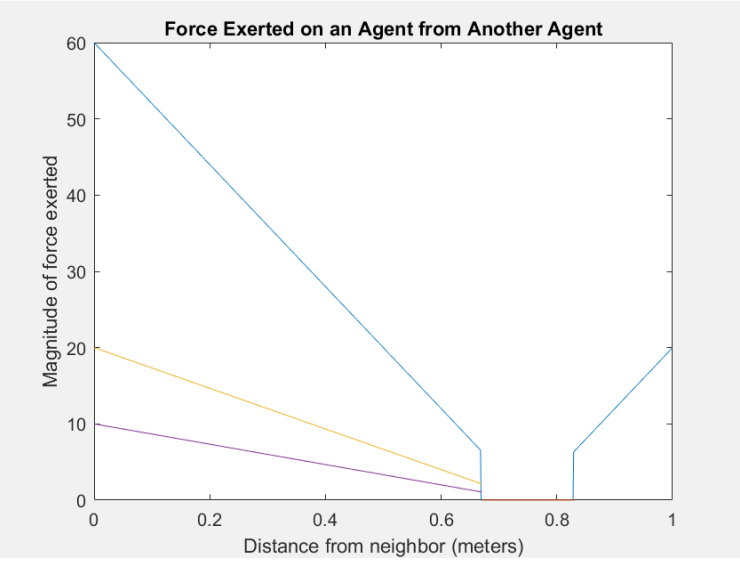


Figure 3.5. Force exerted on the agent from other agents in the swarm. Agents closer to the asset will exert the strongest repulsive and attractive forces (blue line), agents with the same state will exert a medium repulsive force (orange), and agents farther from the asset will only exert a weak repulsive force (purple). $D = 0.75$, $\epsilon = 0.08$

the swarm have the mission of protecting the asset, and the goal is to form barriers around the asset and approach any potential threats with the intention of neutralization.

In order to obtain this behavior, the robot must be drawn to the asset. Granted, they should not be so close so as to impede the asset's movement. As such, the forces derived from the asset imitate the Artificial Physics framework. As depicted in Figure 3.4, the magnitude of the force is dependent on the distance from the asset. Whether the force is attractive or repulsive depends on if the robot is closer or farther than the ideal distance plus or minus some error ($D \pm \epsilon$).

The forces between members of the swarm get slightly more complicated. In an attempt to replicate the behavior of the SPH approach, each robot still contains a state that signifies its distance from the asset. For each neighbor with a lower state (ie: closer to the asset), the robot uses the exact force calculated as if it had seen the asset itself. Using a combination of the attractive and repulsive forces encourages each robot to remain as close as safely possible to the asset. However, for all neighbors that are of the same state or higher, there is no attractive force. This is to keep the surrounding neighbors from pulling the robot closer to the outskirts of the swarm and keeping it from following the asset once it becomes mobile. The combination of forces exerted on a member of the swarm by other agents is shown in Figure 3.5. As one can see from the figure, the strength in magnitudes is different as well, providing extra assurance that those neighbors closest to the asset have the most influence on the robot's velocity.

The last influence on a robot in this method is a threat. Like assets, threats exert both a push and a pull force on each agent. The pull force guarantees that some agents will address the threat while the push will keep it from colliding. It should be noted as

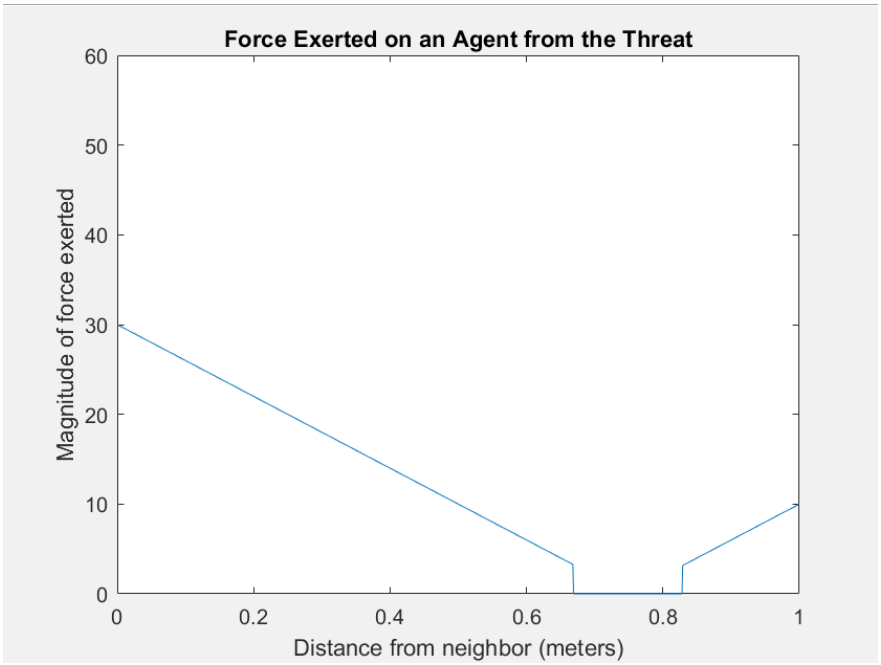


Figure 3.6. Attractive and repulsive force exerted on an agent by a threat. (D = 0.75 and $\epsilon = 0.08$)

well by Figure 3.6 that the strength of this force is not to exceed the strength of the attraction to neighbors with lower states, as a single threat would be able to draw a member of the swarm away. Given enough time, an intelligent threat would be able to lure all robots away and neutralize them, diminishing the effectiveness of the overall system.

3.2.2 Experiments

The following is an outline explaining the setup and purpose of each of the experiments run on the SPH and the potential field based approaches. Unless otherwise stated in the explanation of the experiment, all variables and algorithm parameters are defaulted to the list presented in Table 3.3. These parameters have been determined

Parameter	Default Value	
Maximum Robot Speed	10.0	cm/s
Maximum Distance a Neighbor is Detectable	100.0	cm
Maximum Distance for State Change	50.0	cm
Distance Epsilon	8.0	cm
Smoothing Length	150.0	cm
Surface Tension Coefficient	1.0	
Adhesion Coefficient	-10.0	
Viscosity Coefficient	0.01	
Particle Mass Coefficient	1.0	
Resting Fluid Density Coefficient	1.0	

Table 3.3. Default parameters for all experiments

heuristically to cause both methods to behave as ideally as possible while still maintaining enough similarity to be comparable.

Experiment 1 - Stable Formation Test:

The first experiment to be run on the proposed algorithm is intended to test the scalability and overall function of the system to establish a stable formation around the asset. A stable formation is defined in this instance to be one in which no more than 10% of the population of the swarm switches states (including switching between layers of liquid states) within a period of 10 seconds. The concept of a stable formation is relatively important in both the SPH and the potential fields based methods due to the way that both algorithms determine velocity of a single agent. As previously mentioned, robots in the swarm are influenced by surrounding robots. Hence, it is possible in a compact system for the inter-swarm forces to overpower the obstacle avoidance forces and cause a robot to crash into an obstacle or even the asset itself. Once the system is

stable, however, the asset will be able to move freely without concern of collision with agents of the swarm due to the minimization of neighbors and hence a reduction in the likelihood of inter-swarm interactions overpowering the obstacle avoidance forces.

At the beginning of this experiment, a number of agents (starting with 10 and increasing up to 150) will be placed in random locations uniformly around the asset. However, the density of the initial configuration will be considerably high, causing the swarm to have to expand into a normal, stable configuration. Each experiment will be run 10 times, and the average time it takes to obtain a stable formation will be recorded. These stable formation times of varying sizes of swarms will show the scalability of the proposed algorithm.

Experiment 2 - Offset Initialization Formation:

The next experiment is similar to the first, though the agents of the swarm will not be placed uniformly around the asset. All members of the swarm will be placed to the lower left of the asset such that at least one member of the swarm can sense it nearby. Images of the formation at specific intervals will be provided for vary sizes of swarms including 10, 50, and 100 robots. This experiment will prove the ability of the SPH based algorithm to form a useful, protective barrier around the asset despite the initial positioning of the members of the swarm.

Experiment 3 - Mobile Asset:

While obtaining a stable formation is a challenging and useful function by itself, the goal of this work is to provide protection to an asset while minimally restricting its

movement. The next experiment then, will test both systems' ability to not only guard a mobile asset but also to reconstruct the original stable formation after the asset stops moving. It is expected that the formation will change as the asset moves, and thus this experiment will reveal the system's mobile formation along with the its resilience.

In terms of movement of the asset, the worst case scenario for either system is that the asset continues moving in a straight line for an extended period of time. As such, this experiment will be conducted on three sizes of swarms - 10, 50, and 100 agents - and monitor the movement of the swarm as the asset becomes mobile for 5 minutes and then stops. The system will be allowed to reach a stable formation as defined in the first experiment before the asset begins movement.

Experiment 4 - Threat Pull:

Another goal of this work is to allow the agents of the system to address a potential threat if one is identified, thus both methods have some attractive force to any threats. While that allows agents in the swarm to neutralize any threat to the asset before it even approaches, it also allows for the threats to potentially draw members of the swarm away from the system. Even in cases such that the pull of a threat is lower than the pull force of other members of the swarm, if multiple threats appear in the same area, the force generated may overpower the force that keeps the agent in the swarm, and coordinated efforts of the threat may be enough to draw a select few robots away.

To test this, varying numbers of threats will slowly approach a single swarm of adequate size in a stable formation around a static asset. Upon getting within sensing distance of the outermost robots in the swarm, the threats will retreat. An agent will be

considered to be pulled away if it shifts states in direct response to the introduction of the threat.

Experiment 5 - Complete Simulation:

Finally, a complete simulation will be run involving a mobile asset and multiple mobile threats. An asset will attempt to get from one end of a field to the other while multiple threats cross its path and address the swarm protecting it. Pictures of the result will be provided in section 4.1 and discussed in section 4.2.

Chapter 4

Results

As previously mentioned, in order to prove the usefulness of the proposed algorithm, each experiment will be run twice - once on the SPH based control algorithm, and once on the more common method of using potential fields. The first section will solely provide the results and explain some of the finer points of each experiment while section 4.2 will discuss the implications of the data collected. Both sections will be split based on the experiment in question. The last section will discuss the limitations of the proposed solution.

4.1 Data

4.1.1 Stable Formation

The definition of a ‘stable formation’ used in this work is one in which no more than ten percent of the population of the swarm switches states within ten seconds as defined in section 3.2.2. This is a count per robot, meaning that one robot is allowed to switch states as many times as possible and still be considered as only 1 change within that ten second period. In addition, the time it takes to reach the stable formation does not include the ten second period itself. Hence, for example, after the formations in Figures 4.1 and 4.2 are reached by the SPH based algorithm and the potential field based

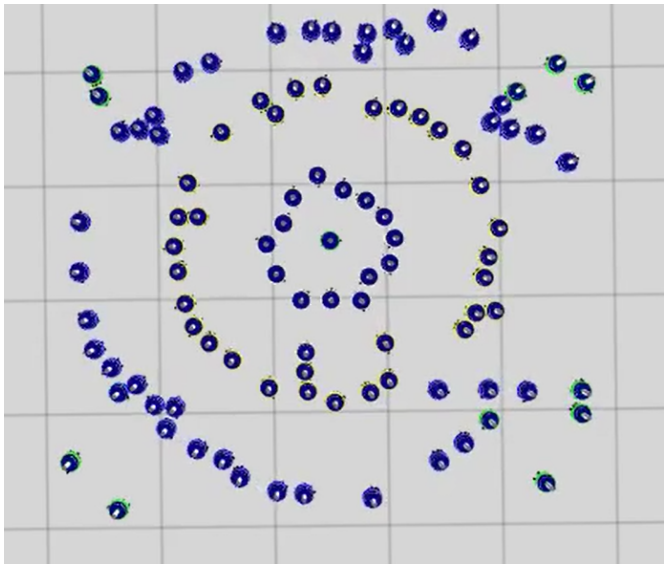


Figure 4.1. The moment a stable formation is achieved using the SPH based control algorithm on a swarm of 100 robots. This formation was achieved after 7.8 seconds with a maximum speed of 10 centimeters per second.

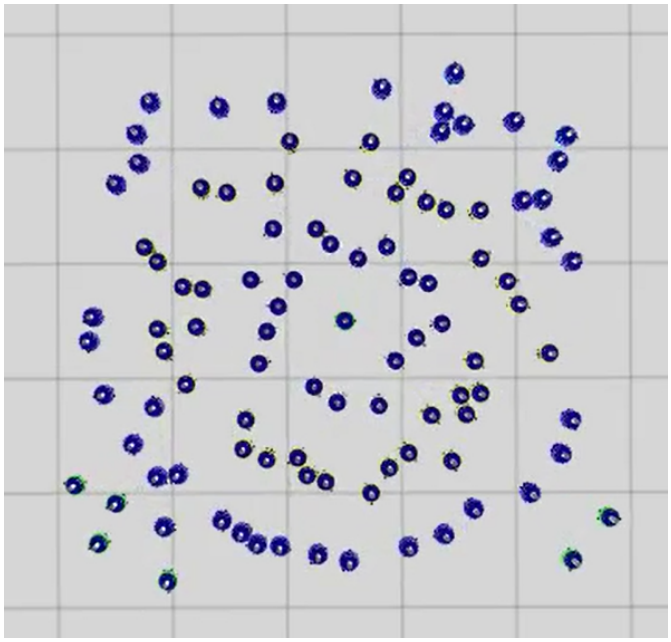


Figure 4.2. The moment a stable formation is achieved using the Potential Field based control algorithm on a swarm of 100 robots. This formation was achieved after 1.7 seconds with a maximum speed of 10 centimeters per second.

algorithm respectively, no more than ten robots switch states within the next 10 seconds. Both algorithms start with the same exact initial (random) positions of the swarm around the asset.

In order to reduce the possibility of skewed data in which a random position of robots may be a better configuration for the start of one method over the other, 10 trials of each are run, each with a different random position of the robots scattered around the asset. The individual times of each trial is represented on the graph in Figure 4.3. The average of all times is plotted in Figure 4.4, revealing a similar trend between both methods.

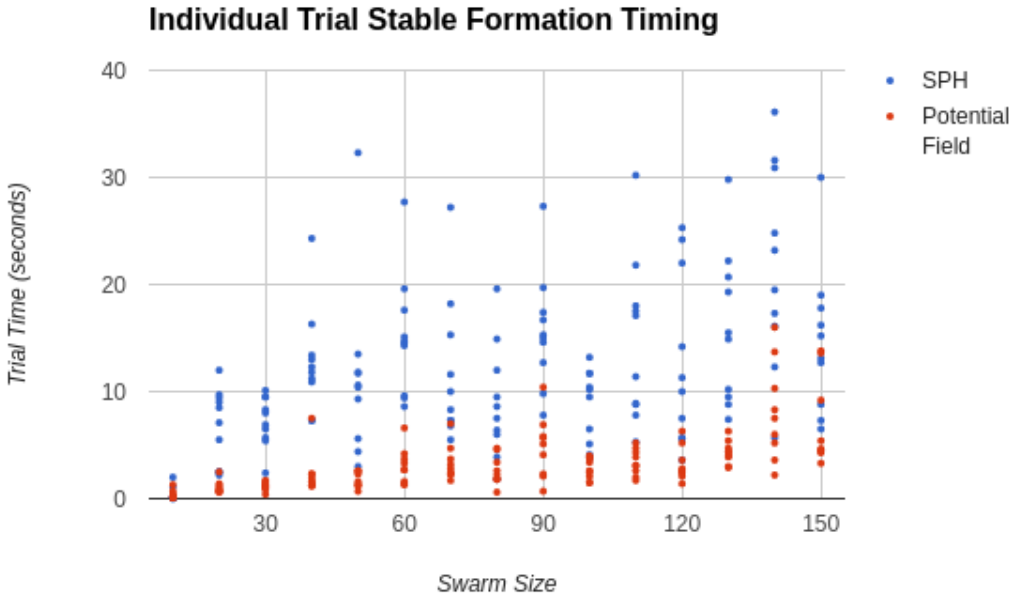


Figure 4.3. The time it takes for swarms of various sizes to reach a stable formation using both the proposed SPH based control algorithm and the alternate Potential Field based algorithm.

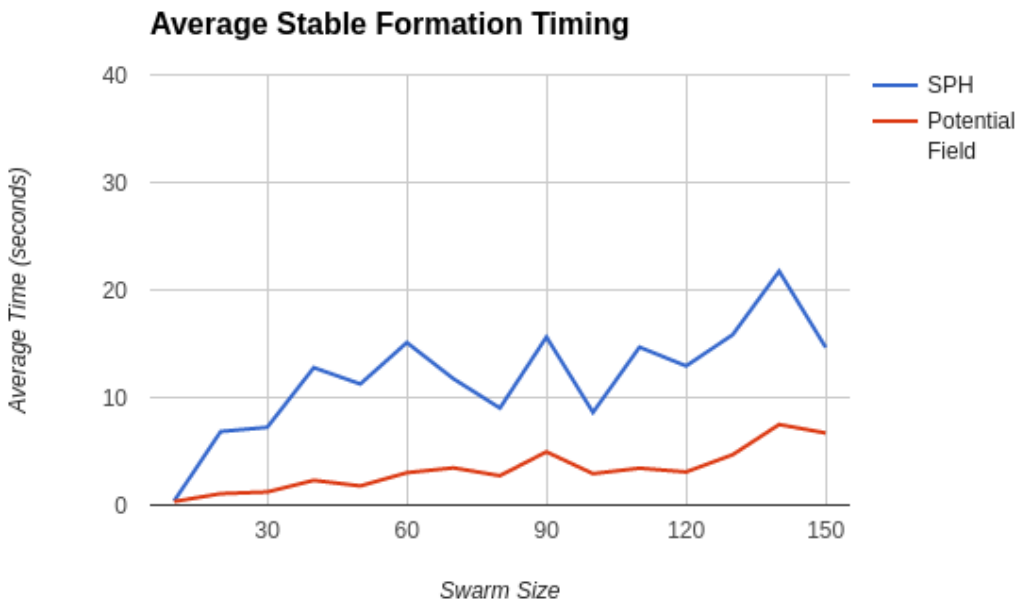
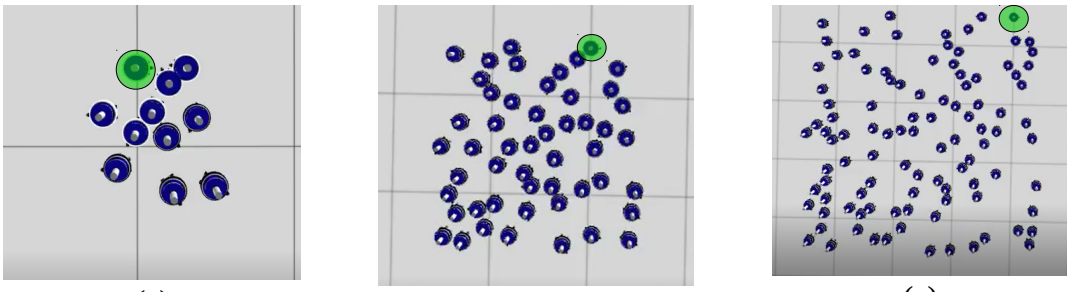


Figure 4.4. The average time it takes for swarms of various sizes to reach a stable formation using both the proposed SPH based control algorithm and the alternate Potential Field based algorithm.

4.1.2 Offset Initialization

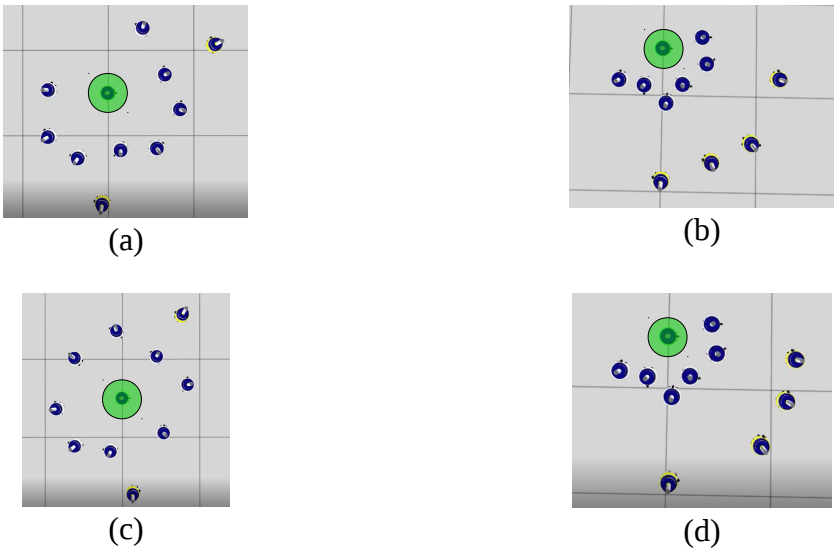
Of course, the data collected from the first experiment comes from an initialized position in which the swarm is released relatively uniformly around the asset in a compact state. However, in practice, odds are that the swarm will be released all at once near the asset, not around the asset. Hence, this experiment tests the ability of both functions to form a suitable formation given an initial position of the swarm that has some slight offset. Three different sizes of swarms have been tested: a 10 robot swarm, a 50 robot swarm, and a 100 robot swarm. The initial positions for the three swarms are shown in Figure 4.5, where the robot circled in green is the asset the swarm is programmed to protect.



(a) (b) (c)

Figure 4.5. The starting positions for the offset initialization testing for swarms of sizes 10 (a), 50 (b), and 100 (c). The asset is highlighted in green.

As shown in Figure 4.6, when using the SPH based control algorithm, the swarm reaches close to its final formation in roughly 27 seconds. After that period of time, the swarm's formation changes very minimally, though the swarm does not manage to completely surround the asset. When using the potential fields based method, however, the swarm takes roughly a minute to reach the point where minimal change will occur, though the swarm does completely surround the asset.



(a) (b) (c) (d)

Figure 4.6. The formation of the 10 robot swarm around an asset (green) at 27.5 seconds using the Potential Field based algorithm (a) and the SPH based algorithm (b), then again at 60 seconds using the Potential Fields algorithm (c) and SPH (d).

The results for the swarm made of 50 robots, however, is slightly different as one can see in Figure 4.7. Again, it takes roughly a minute for the innermost layer of robots to surround the asset when using the potential field based algorithm, though once that ring does form, the rest of the swarm begins spiraling around the asset in a counterclockwise manner. After 13 minutes, the second layer of robots - almost completely surrounds the first, though it doesn't quite manage to complete within that time. The spiraling is unique to the potential fields method, however, as the SPH based algorithm results in the swarm enclosing the asset from both sides. The innermost layer never completes, leaving a small gap opposite of the side the swarm started, though the second layer seems to form a decent barrier similar to that of the potential fields based algorithm after 13 minutes as depicted in Figure 4.7 (j).

Finally figures 4.8 and 4.9 show the results of the 100 robot swarm using the potential field based method and the SPH based method respectively. The results are similar to that of 50 robot swarm, where the innermost ring of the potential field swarm forms relatively quickly (under 2 minutes) and as soon as it does, begins to spiral around the asset. After roughly 7 minutes, the swarm finally begins to close around the asset, forming a more evenly distributed system. However, the density still does not even out until a few minutes later, considering at any one point, half of the swarm contains significantly fewer robots than the other half. Also like the 50 robot swarm, the 100 robot SPH based swarm does not spiral and instead encloses the asset from both directions. After five minutes, as shown in Figure 4.9 (e), the innermost liquid state layer is formed. The second layer completely surrounds the asset about two minutes later and eventually evens out the distribution of the rest of the robots.

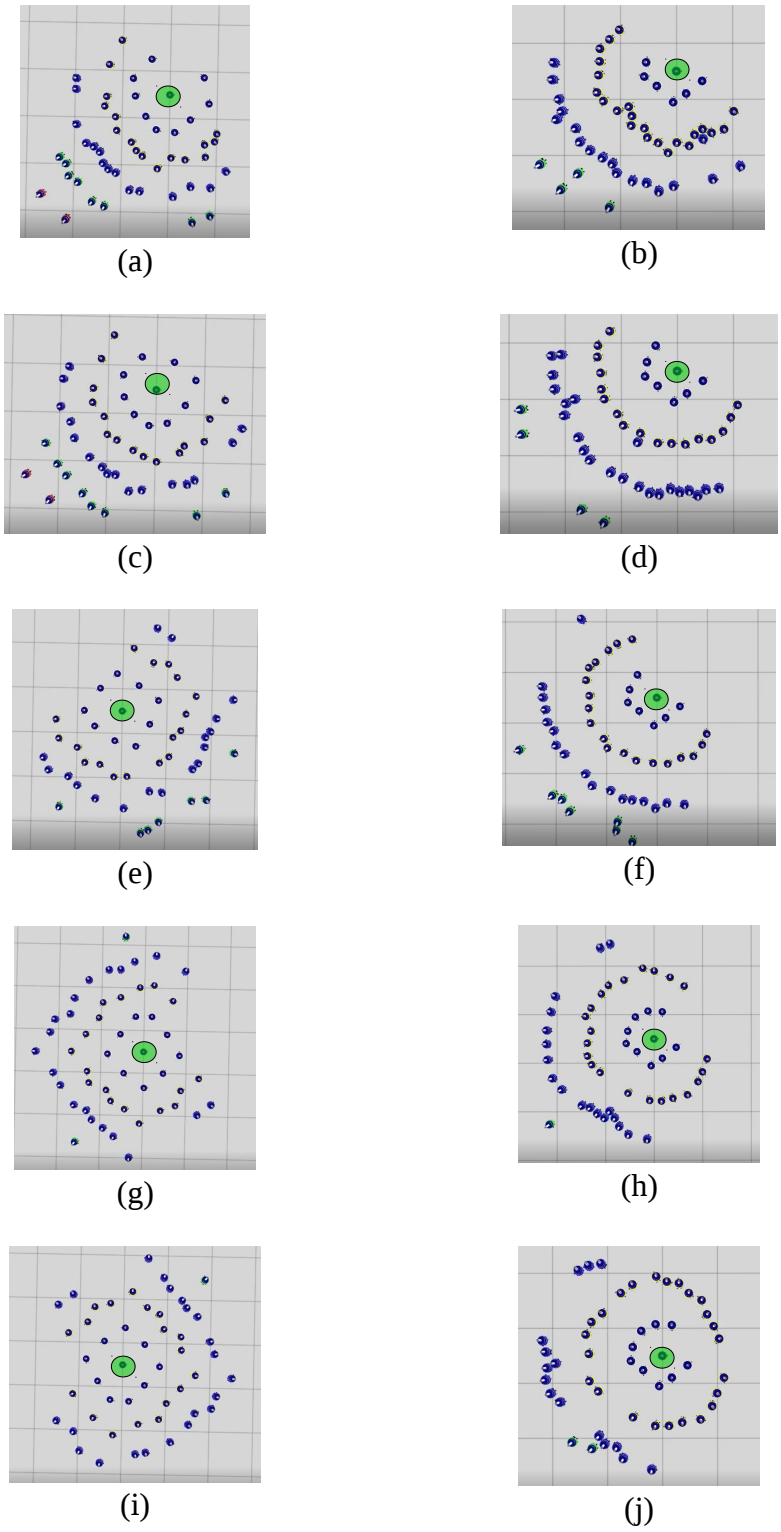


Figure 4.7. The formation of the 50 robot swarm at 0.5 minutes [PF – (a), SPH – (b)], 1 minute [PF – (c), SPH – (d)], 3 minutes [PF – (e), SPH – (f)], 9 minutes [PF – (g), SPH – (h)], and 13 minutes [PF – (i), SPH – (j)].

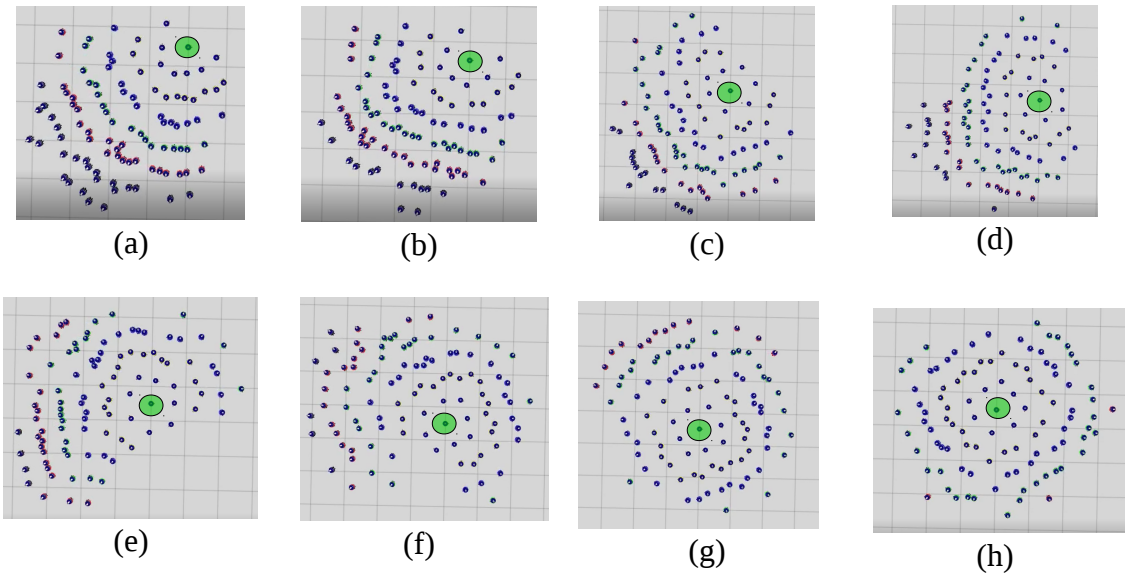


Figure 4.8. The formation of the 100 robot swarm using the Potential Field based algorithm at 0.5 minutes (a), 1 minute (b), 2 minutes (c), 3 minutes (d), 5 minutes (e), 7 minutes (f), 9 minutes (g), 13 minutes (h)

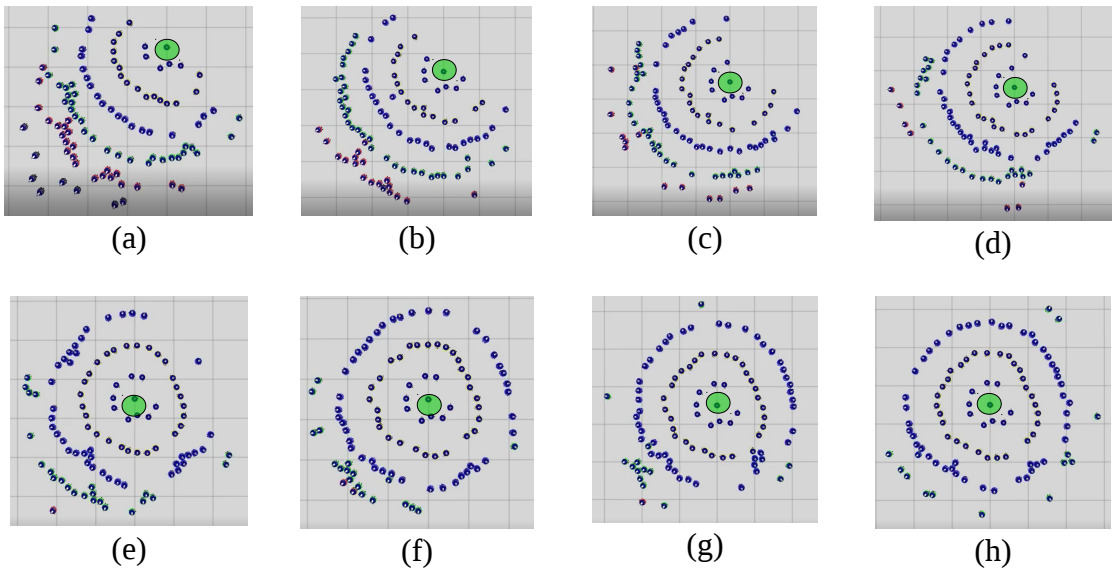


Figure 4.9. The formation of the 100 robot swarm using the SPH based algorithm at 0.5 minutes (a), 1 minute (b), 2 minutes (c), 3 minutes (d), 5 minutes (e), 7 minutes (f), 9 minutes (g), and 13 minutes (h).

4.1.3 Mobile Asset

The results of the third experiment in which the swarm is tasked with following a mobile asset without impeding its movement follows the patterns set forth by the second experiment. A swarm of 10 robots proves to be functional in terms of the potential field algorithm, but not the SPH based algorithm. After allowing the swarm to reach a stable configuration such as those in Figure 4.6 (c) and (d), the potential field kept the exact formation the entire time the asset was mobile. For the SPH based algorithm, the swarm merely shifted such that the cluster of robots trailed behind the asset, still following it, but not surrounding it.

The 50 and 100 robot swarms, however, were much more interesting. The results from both were almost identical, and as such only the 100 robot swarm is shown in Figures 4.10 and 4.11. When using potential fields, the formation holds relatively well around the asset. However, the density of the robots in the swarm slowly increases behind the asset and decreases in front. This leads to a break in the second layer of robots after three minutes of moving and a thinning of robots in front of the asset in the first liquid layer of robots after 5 minutes.

The same type of behavior is observed in the swarm using the SPH based control algorithm as well, though more drastically. After one minute, there's a break in the outermost layer of the swarm's barrier in the direction the asset is moving as depicted in Figure 4.11 (b). By the second minute, there's a break in the next layer and more robots begin to trail behind the asset. No robots are 'lost' in that all continue to follow the asset, and the innermost layer of robots never seems to break from the SPH based swarm as shown by Figure 4.11. Overall, the robots in the SPH based method form much more

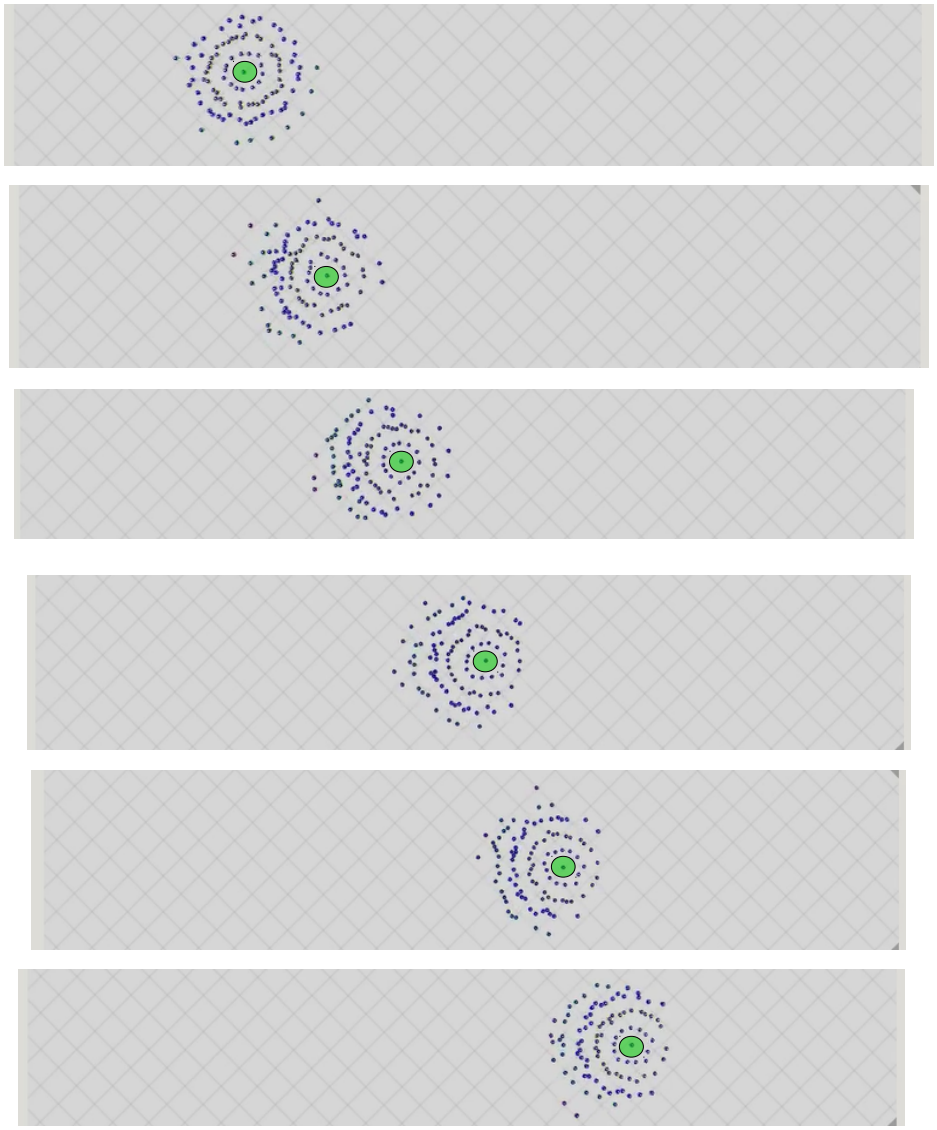


Figure 4.10. The formation of the 100 robot swarm using the Potential Fields method as the asset begins moving (a), and after it has been moving for 1 minute (b), 2 minutes (c), 3 minutes (d), 4 minutes (e), and as soon as it stops after 5 minutes (f).

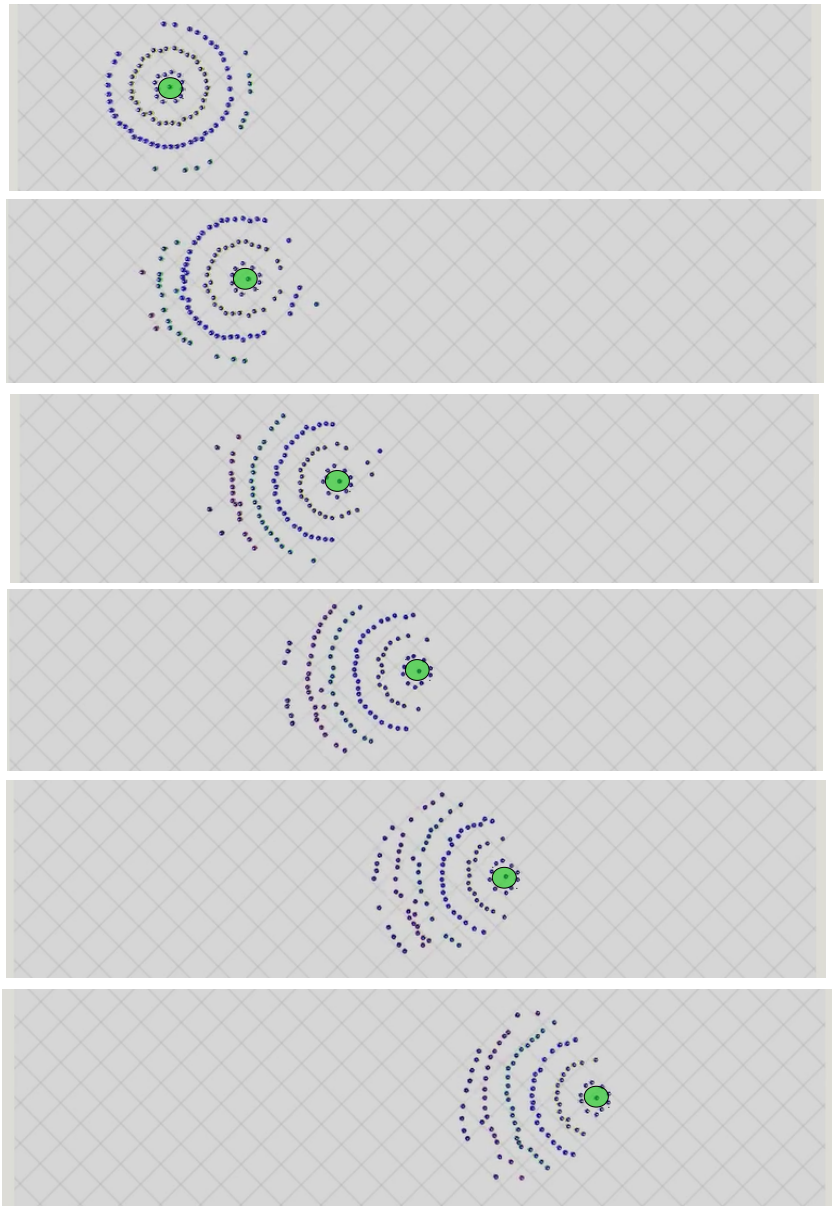
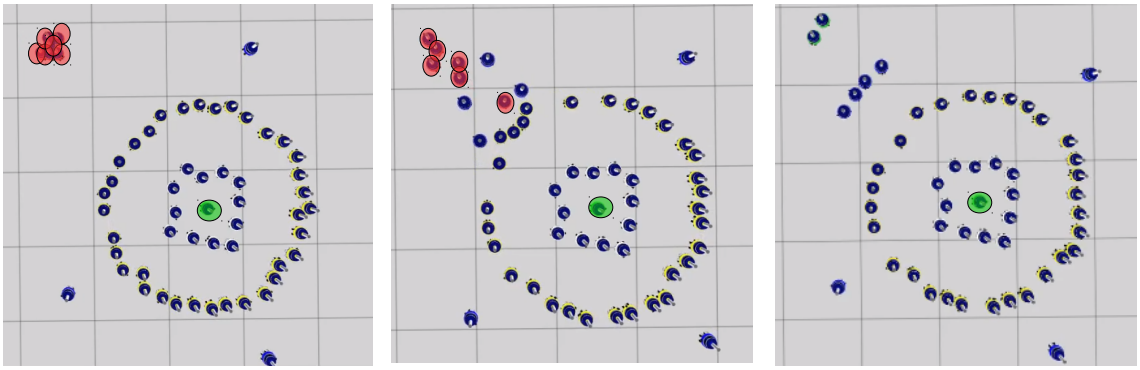


Figure 4.11. The formation of the 100 robot swarm using the SPH method as the asset begins moving (a), and after it has been moving for 1 minute (b), 2 minutes (c), 3 minutes (d), 4 minutes (e), and as soon as it stops after 5 minutes (f).



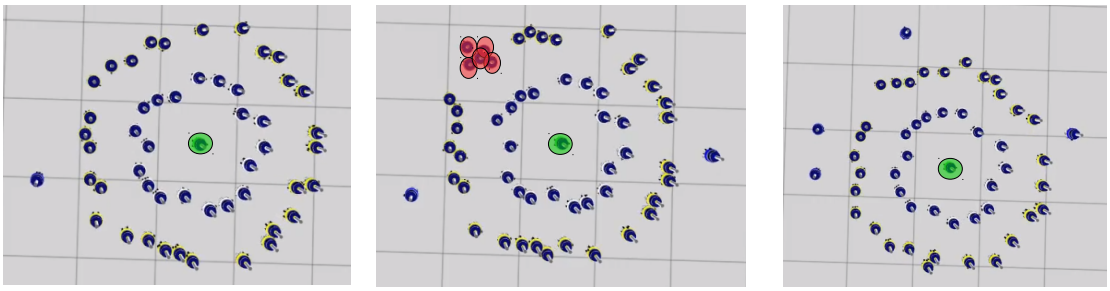
(a) (b) (c)

Figure 4.12. A swarm of 50 robots using the SPH based algorithm in its initial state (a), then reacting to 6 threats (highlighted in red) (b), and the final formation after the threats leave (c).

structured and organized layers of robots behind the asset as compared to the clusters of agents that form in the potential field based swarm.

4.1.4 Threat Pull

The fourth experiment tests the functions' abilities to address a threat while still keeping a reasonable barrier around the asset. A swarm of 50 robots is allowed to reach a stable formation before any number of threats is introduced into the scene. Figure 4.12 shows the key parts of the SPH based algorithm when attempting to protect the asset from six threats. The threats approach from the upper left in a group. Once the threats reach the edge of the outermost layer of robots, they begin retreating back from where they came. As one can see in Figure 4.12 (b), nine robots responded in some way to the threat, whether by following after it a short distance or gathering closer to other robots between the threat and the asset. In the end, six robots were pulled outward, four to the second layer, and two to the third.



(a) (b) (c)

Figure 4.13. A swarm of 50 robots using the Potential Field based algorithm in its initial state (a), then reacting to 5 threats (b), and the final formation after the threats leave (c).

The potential field algorithm reacts in a similar manner. Figure 4.13 (b) shows nine robots reacting to the five incoming threats. Though instead of gathering together, the repulsive forces make nine robots part and break the barrier. Though when the threats retreat, four robots pull outward, two robots from the innermost ring move to the outer ring, and two move from the outer ring to help form a new ring as shown in Figure 4.13 (c). The number of robots that react per volume of threats is plotted in Figure 4.14, while the number of robots that are pulled outward is shown in Figure 4.15.

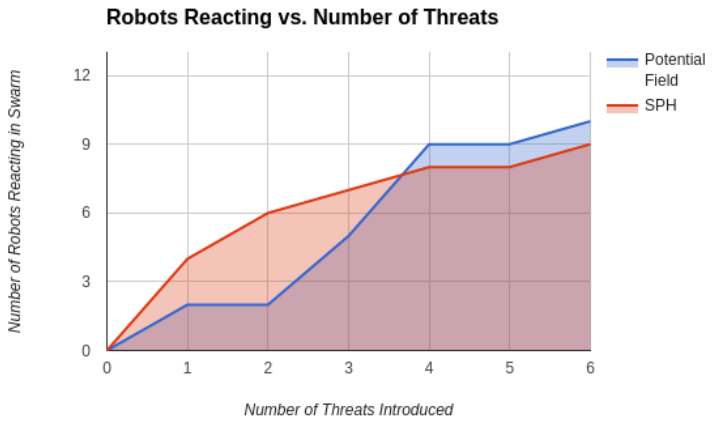


Figure 4.14. A graph of the number of agents in the swarm that react in some manner to a particular number of threats.

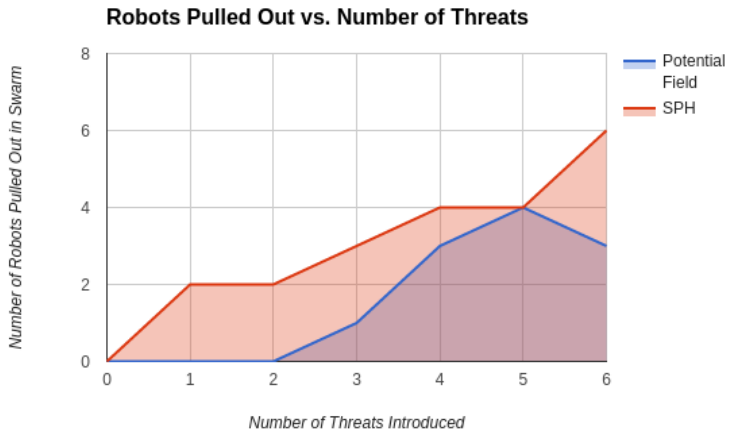


Figure 4.15. A graph of the number of agents in the swarm that are pulled into outer layers in response to a particular number of threats.

4.1.5 Complete Simulation

Finally, to prove the effectiveness of the algorithms in a complex environment and to reinforce the principles discovered previously, a simulation was run in which a swarm of 100 robots protect a mobile asset from three mobile threats as the asset moves along. Three mobile threats moved back and forth while an asset attempted to navigate through them. The trajectory of the asset was set manually so that the swarm would have to encounter the mobile threats. Figure 4.16 shows the progression of the SPH based algorithm upon which the swarm encounters each threat. In every situation, the robots closest to the threat cluster together as in the fourth experiment to neutralize the threat. The same, familiar trailing found in experiment 3 reappears as the asset moves, and as soon as it stops, the swarm regroups around the asset as presented in the second experiment. No new data was discovered, rather it has been validated that the behaviors

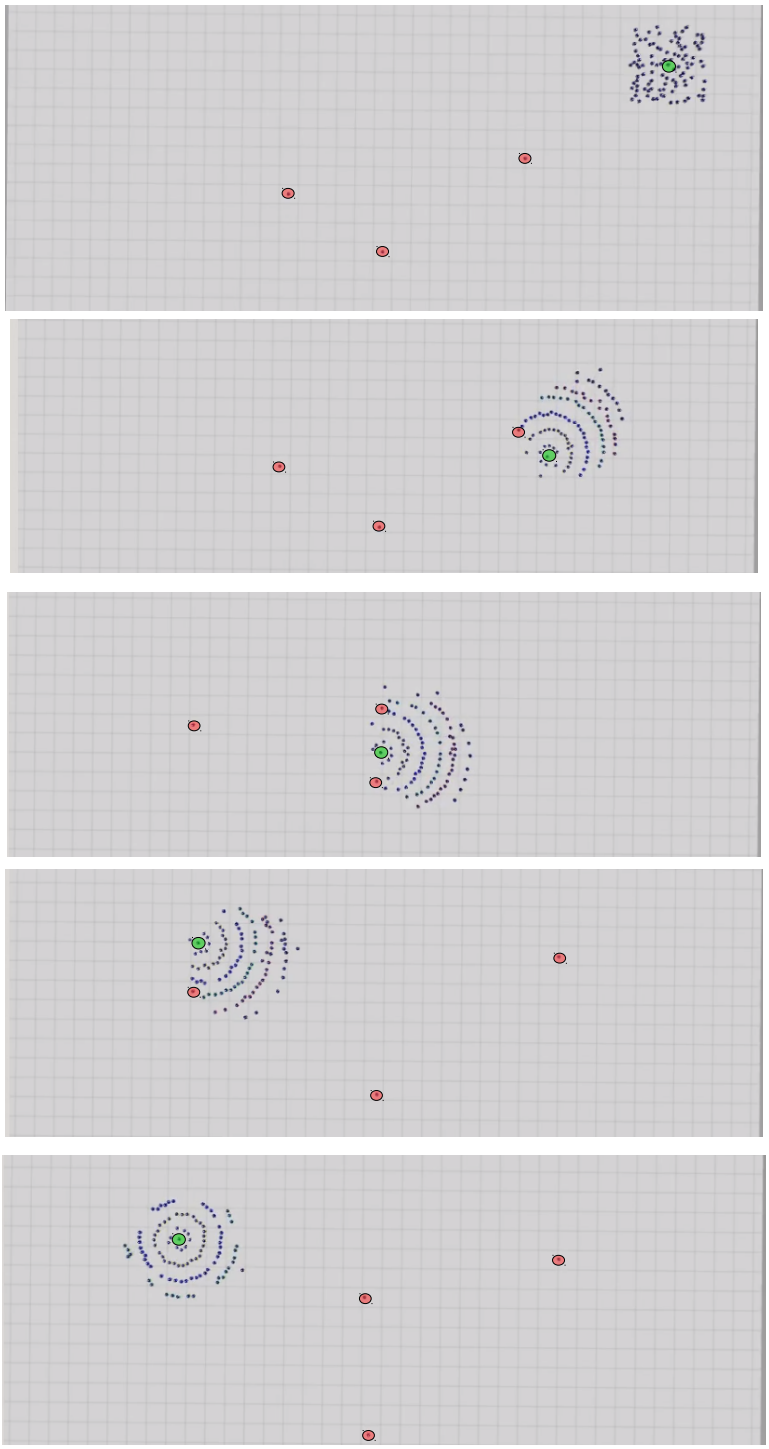


Figure 4.16. A complete simulation of an asset moving around three mobile threats.

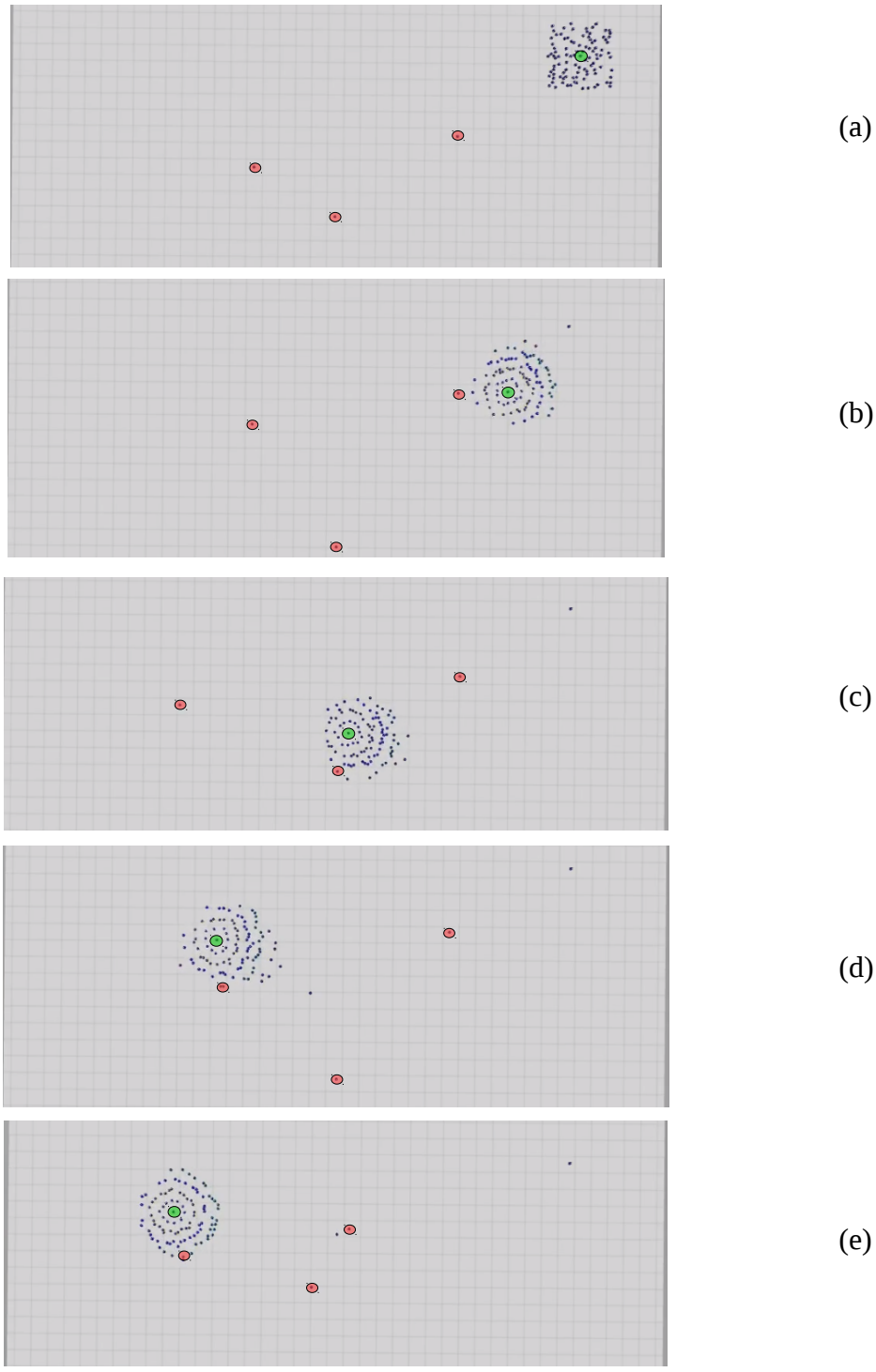


Figure 4.17. A complete simulation of a potential field based swarm protecting a mobile asset from mobile threats.

observed in the SPH algorithms are consistent even in more complex environments.

Figure 4.17 shows the same relative success with the potential field based algorithm. The swarm members exhibit the same behavior including the trailing affect, the slight clustering of agents, and the redistribution of robots when the asset stops as observed in previous experiments. There is some new data, however, in that throughout the course of the experiment, two robots break away from the swarm and thus the volume of the swarm around the asset at the end is decreased slightly.

4.2 Discussion

4.2.1 Stable Formation

The first experiment reveals quite a bit about the differences between the SPH based and potential fields based solutions presented in Chapter 3. The stable formations presented in Figures 4.1 and 4.2 show that the overall structure of both are inherently different. SPH forms a uniform, ordered layer of rings around the asset, while it is the nature of potential fields to create more of a lattice structure where all robots are equally distant from other robots and the asset. It is this inherent behavior that causes the difference in range of results presented in Figure 4.3. According to the scatter plot, the SPH based algorithm seems to be much more volatile - considerably more dependent on the initial position of the swarm than its potential field counterpart. Even with only 50 robots in the swarm, one trial may end in a stable formation found after less than 5 seconds while another may be found after 30. However, after considering the behavior of the system and how the robots interact, this concept makes sense.

Taking the swarm in Figure 3.2 for example, even though the rings of robots around the asset are spaced out relatively far from other rings, robots within the same layer are very close. As the robots communicate via infrared technology, if two robots are communicating and another robot moves into the line of sight, the communication between the original two robots is broken. Hence, if the already close robots attempt to move closer together such that one or more move in front of other members of the same layer and block its communication with the robots closer to the asset, that robot will shift states so that it is now a member of the outer fluid. This happens frequently in the SPH based algorithm. However, the potential fields based algorithm desires to form a lattice structure where members are not close to each other. Therefore, it is very difficult for another robot to suddenly block communication and this behavior to occur.

Even though the very natures of these two methods are different, the data presented in Figure 4.3 still proves valuable. A stable state as defined in Section 3.2.2 is not completely necessary to allow mobility to the asset, rather it merely guarantees it. As the innermost layer of protection is the first to form during initialization, the asset will quickly have some ability to move. The trend of the average times presented in Figure 4.4 then is best used to show the scalability of the proposed algorithm, revealing that it may be slightly slower than existing solutions, but it still grows at a similar rate relative to the size of the swarm.

4.2.2 Offset Initialization

The second experiment shows some more differences between the behaviors of the two algorithms that prove interesting. The first and slightly obvious fact is that the

size of the swarm greatly influences the success of the algorithm. As illustrated by the failure of SPH to adequately surround the asset in Figure 4.6 (b), Smoothed Particle Hydrodynamics works best in a large volume system. Each robot determines its velocity based on parameters of surrounding particles, and the more particles there are, the more life-like the response. Ten agents is not enough to allow SPH to function to its full potential. The potential fields method, however, thrives in small groups since it requires a minimum of one obstacle or neighbor to determine some force to act upon.

The next difference revealed by this experiment is how the swarm behaves when surrounding a target. The potential fields based method results in a spiraling around the asset, slowly filling in gaps as all members circle around as shown in Figure 4.8. However, the SPH based method surrounds the target from both sides as if a fluid crashing into and around an obstacle. There are benefits to both approaches, as swirling around the target can provide a patrol-like behavior. A high density of robots is not stuck to one side of the target for the entire time it takes to completely surround it. However, in the case of protecting a human or an asset capable of protecting itself in one direction, it may be better to have the swarm uniformly engulf the asset from both sides, narrowing the window in one direction such as that shown in Figure 4.9 (c) and (d).

The last and perhaps most intriguing property of the SPH based solution is best illustrated by Figure 4.9 (g). In this situation, the inner liquid layer (yellow robots) formed a small gap in the perimeter, caused by the tightening of the robots elsewhere in the layer. However, this gap in the perimeter is eventually filled. It attracts a large cluster of robots from the outer layer and draws them in closer. Eventually, the robots from the outer layer fill in the gap, creating a property of resilience in the overall system. This

resilience is caused by the surface area minimization discussed previously and calculated using the normals of neighboring particles. As the surface area is less when there is no gap, the robots in the outer layer move closer to fill it, creating a self-fixing structure. This property is also observed by the fact that the robots in the inner liquid layer are not completely circular. The robots closest to the gap in the solid state robots - while not filling the gap - do provide a tighter perimeter and thus a better security at that particular spot.

4.2.3 Mobile Asset

The third experiment reiterates what was learned in the second in relation to the size of the swarm. Once again, SPH fails to succeed in providing adequate protection when the swarm is small. However, in a larger volume swarm, the SPH based algorithm provides a very secure and structured formation. As is evident by Figure 4.11, the robots in front of the asset eventually shift to move directly behind the asset in distinct layers. It is because of this behavior that the results of the second experiment prove useful. Assuming the asset can see and defend itself in one direction, as soon as it stops moving, the robot swarm will begin to surround the asset from both sides. In this manner, the asset only ever has to worry about what lies in front of it.

This behavior differs from that of the potential field based algorithm in that the robots of that particular swarm are much more chaotic and unorganized. Figure 4.10 shows that the potential field based method produces a stronger barrier in front of the asset, but behind the asset, there are random clusters of robots in addition to relatively wide gaps. This unpredictability may be seen as a good thing as any intelligent threats

may not know when or how to penetrate the barrier, but unlike the tight and resilient perimeter formed by the SPH method the asset is not guaranteed to have a relatively uniform distribution of robots protecting it. In addition to this, while the potential fields method keeps some robots in front of the asset, Figure 4.10 shows that the number of robots in front slowly decreases as the asset continues moving. One can argue that if the agent were to continue indefinitely, there would come a point where there would be no members of the swarm in front of the asset. The SPH based algorithm, however, has the innermost, solid state layer which shows no sign of diminishing in Figure 4.11.

4.2.4 Threat Pull

The results of the fourth experiment prove the usefulness of using the adhesion formula expressed in earlier sections as a means of protection. While no actual methods of neutralization were used, it is evident that the system responds properly to the increase in threats by the steady rise in the number of robots that respond to said threats. The trend for the SPH based method is much more steady than that of the potential fields method and most of the time surpasses it in how many robots respond at all. The only reason the number of robots that react in the potential fields method may surpass that of SPH (as seen in Figure 4.14) is because the innermost layer of the SPH method is designed not to pay attention to any threats in an effort to provide a stable, last-defense perimeter around the asset.

Figures 4.12 and 4.13 show one more interesting feature of these methods. Figure 4.12 shows that the SPH based swarm naturally creates additional barriers after a threat has been detected. The six robots that respond to the threats jump out to start to form

additional layers while still remaining a part of the swarm. This can be seen as a benefit to the system as if the threats approach once again from the same direction, the robots can address them before they can get as close to the asset as they once did. If the threats do not come back, the outermost layers eventually succumb to the surface area minimization property of the SPH based algorithm and once again move back to be as close to the asset as possible.

Figure 4.13, though, shows that the potential fields method doesn't necessarily respond in this same manner. Of course, while it is true that the perimeter breaks in Figure 4.13 (b) and the members of the swarm let the threat get closer to the asset, this is merely due to the repulsive forces exerted by the threat and can be changed easily by manipulating the magnitude of the force the threats exert and the ideal distance from the threats. However, Figure 4.13 (c) shows that the two agents that are pulled into the outermost layer spread out instead of remaining around the same location in case the threats come back once again. And in this manner, the SPH algorithm responds more appropriately to introduced threats.

4.2.5 Complete Simulation

The final experiment's most interesting point is the fact that two robots escaped from the swarm when using the potential fields based approach. The possibility of this happening is small, but it is still possible. As the forces between robots generally include attractive and repulsive forces, it could be the case that a robot on the outermost edge of the swarm receives a strong repulsive force from its neighbors to force it far enough away such that it no longer senses the rest of the swarm. Since it no longer senses any

neighbors, there are no more points of interest to exert any force on the robot and the agent merely stops.

Besides that one point, however, the experiment merely succeeds in reinforcing what was learned through the past experiments. Even in a more complex environment, the SPH algorithm still enforces a much more organized, compact structure which clusters together when threats are sensed. The behavior is similar to that of the algorithm using a more commonly accepted technique of potential fields, yet slightly improved. The formation is less chaotic, more resilient, and the robots are more responsive to threats.

4.3 Limitations

Of course, any algorithm has its limitations, and the proposed SPH algorithm is no exception. The first limitation to this algorithm is the speed of the robot. Every experiment run previously assumes that the agents of the swarm is at least twice as fast as the asset. Obviously, if the asset is just as fast as the surrounding robots, the likelihood of robots being left behind increases significantly. If the robots cannot catch up to the asset, the asset is very likely to outrun its own protection.

The next limitation is the trailing problem shown in Figure 4.11 where the majority of the robots form behind the mobile asset. The problem is caused by the surface minimization equation of the SPH method. The robots in front of the asset act upon a force similar to that of the normal vector. This causes the robots in front to eventually spread out. The robots directly behind the asset, however, are pulled closer to the asset, acting upon a force similar to the inverse of the normal vector. This forces the robots

behind the asset to attempt to reach the same relative position, and eventually one robot wins over the others. The winning robot cuts in front of the losing robot, blocking the line of sight between the losing robot and the asset, and forcing the losing robot to shift back to an outer layer. This compacting of the robots immediately behind the asset continues, decreasing the numbers there while the spreading of the robots in front of the asset eventually forms a break in the perimeter and the same trailing effect that is observed in Figure 4.11. It should be noted that similar behavior is also seen in the potential fields based method, though not quite as drastically since the SPH algorithm relies explicitly on the normal vector. Unfortunately, there is no obvious way to solve this problem, as reducing the dependency on the liquid's normal vector would also reduce the resiliency of the system which is dependent on the surface minimization equation, and therefore the normal vector. This may potentially be the proposed solution's largest problem.

The last major limitation is the communication between robots. Unfortunately, calculation of inter-particle forces requires knowledge of surrounding particles that are not easily calculated by mere sensing. Hence, the communication capabilities of each robot limit the functionality. There needs to be enough ability to communicate the velocity vector, normal vector, and particle density. In two dimensions, this is represented in five floats. In three dimensions, it would require seven floats. Factoring in another byte or two to represent the state of the robot would also increase the data communicated. As of the algorithm outlined in this work, a robot's state is communicated by a single byte. This means that there are only 256 states available for a member of the swarm. One state must be the gaseous state and another must be the solid state. This leaves only 254 possible sub-states for a liquid robot. Fortunately, at any reasonable volume for a swarm,

this will be more than enough to fully allow all members of the swarm to communicate efficiently.

Chapter 5

Conclusion

While both algorithms described in this work have benefits and disadvantages, the results in Chapter 4 show that Smoothed Particle Hydrodynamics succeeds as a possible solution to the problem of asset protection with a high volume robot swarm. The advantages of the SPH approach seem to outweigh the disadvantages and allow for a very natural and smooth flow of robots around a target. While potential fields are more commonly used for controlling robots and are considerably easier to implement and calculate, SPH provides a very similar and more resilient means for decentralized swarm control.

5.1 Summary

This work relies heavily on concepts and techniques developed by many researchers over the past decades. The concept of potential fields - while simple in idea and calculation - plays a very powerful role in the world of robotics in general. However, a less commonly used technique - Smoothed Particle Hydrodynamics – theoretically would work well to control a high volume swarm. In this calculation, a continuous fluid is broken into discrete particles and analyzed through determining the velocities of the

individual particles. If the swarm is seen as a liquid in which all members are particles and must stay relatively close together, then a fluid model would provide decent estimations of certain properties. These properties can be further customized by adding in a cohesion and adhesion term to better control how the robots in the swarm react to outside entities as well as other agents.

The algorithm proposed utilizes this enhanced SPH method. By using basic line-of-sight communication via infrared technology, each robot in the swarm can communicate its particle density, state, velocity vector, and normal vector. By utilizing various states, a robot is guaranteed to always focus inward towards the asset it is protecting. Overall, this algorithm seems to result in a dynamic and versatile swarm, providing a tighter barrier around the asset while allowing agents to adapt to the introduction of threats into the environment. However, in order to prove the viability of the system in relation to methods already in existence, another alternate algorithm was developed in which the system of robots uses only potential fields to determine velocity. Both were designed to act as similarly as possible in order to achieve a decent comparison.

The first of five experiments established a trend in the amount of time each algorithm achieves a stable formation. The second experiment was then run to prove the algorithm’s ability to form a protective barrier despite not starting in a relatively uniform distribution around the asset. The third tested the swarms ability to follow a mobile asset and keep a protective formation while the fourth was designed to test the ability to address potential threats. Finally, the last experiment merely tested both of the swarms in

a complex environment including a mobile asset and multiple mobile threats to prove that the behaviors observed in previous experiments where properties of the algorithms.

Overall, both functions worked similarly and succeeded in basic asset protection. The SPH based algorithm, though, ended up with a higher density of robots around a static asset and a smaller escape window than the potential fields algorithm as evident by Figure 4.8 (h) and Figure 4.9 (h). The potential fields based function may be more useful for certain scenarios such as that where a large density of robots is needed in front of the asset. Figure 4.10 shows how the shift of agents from in front of the asset to behind it is considerably slower than that of the SPH based method in Figure 4.11. However, in situations where the asset can guard in one direction and only needs the swarm to protect the sides and rear, the SPH may be better suited as a solution. Figures 4.10 and 4.11 also show that the potential field based method results in a high density of agents directly behind the asset, resulting in clusters and potentially more collisions. The SPH algorithm, however, results in relatively evenly spaced rows, resulting in less collisions.

In the end, the better algorithm depends on the specific requirements of the situation. This work does show, however, that the Smoothed Particle Hydrodynamics technique is a potential solution to the high volume asset protection problem and should be considered seriously for further research and analysis.

5.2 Future Work

While this work provides a good basis and proof of concept for SPH based control algorithms, there is still much that may be done to extend this work. The first and most useful extension would be to continue this work into the third dimension. Smoothed

Particle Hydrodynamics was developed for natural, three dimensional simulation, and thus using it to control robots on a flat plane is limiting the potential of the algorithm. Fortunately, the Swarmanoid project includes an ‘eyebot’ as well which has the capability of flight. Now that it is known that this method works well as a control for two dimensional robots, the next step would be to use these flying eyebots instead of the footbots and test the capabilities of the three dimensional swarm. The same simulation environment can be used and the algorithm only needs to be changed to allow for three-term-long vectors instead of two.

Of course, that would also require increasing the communication capabilities of all robots. As previously mentioned, the SPH algorithm requires communication of the velocity and normal vectors. In two dimensions, this means that each vector only has two terms. When simplified down to two bytes, a robot is just barely able to communicate all necessary information within the ten bytes it is allowed to transmit. One possible solution to this problem is to chain the communications. As there are ten iterations happening every second, a robot can simply send half of the data in one iteration and the other half in the next. SPH calculations would then be done every other iteration - or rather, five times a second instead of ten. Theoretically, this should not pose a problem as the frequency of the calculation is still high, though it may be beneficial to determine a better communication method as well.

In relation to the flying robots, another factor that may be important in future research may be stabilization and feedback control. As this technique has been applied to wheeled robots in simulation only, one would expect that there would be a multitude of problems not yet encountered. One very possible issue would the introduction of wind

into the system. In a wheeled robot scenario, wind is a minimal factor as it often is not strong enough to overcome the friction between the wheels and the ground. However, when working with flying robots - especially those of a considerably smaller size and hence more susceptible to wind - one must be able to counteract the forces of external stimuli, and factoring in an efficient method of stabilization into the proposed SPH algorithm may prove slightly challenging.

Taking this one step further would be implementing the SPH based solution in physical robots rather than simulation. The introduction of noise in various sensors are sure to introduce a few interesting outcomes. Unfortunately, as results of this work reveal, SPH is most efficient in high volume swarms, and larger swarms mean higher financial cost. Nevertheless, provided the money was available, verification that this control system works in real life as well as in simulation would prove valuable to this research.

Finally, the last potential extension of this work to be addressed would be that of the introduction of obstacles. As mentioned in Chapter 2, previous researchers have used SPH for navigational purposes with a decentralized swarm in a complex environment. The obstacles in the environment were treated as virtual particles in the fluid, allowing the robots to calculate repulsive forces and avoid obstacles efficiently. The proposed algorithm for asset protection in this work has not addressed obstacles or complex environments at all. And while it has addressed concerns specifically related to the problem statement including mobile threats and a mobile asset, it would benefit from future work involving uneven terrain and obstacle avoidance.

Fortunately, while there are many different areas for improvement in relation to future work, the research done has proven to be successful. A control system based upon

fluid mechanics was established to aid a decentralized, large scale swarm to efficiently and robustly defend an asset from oncoming threats. This system has proven to be comparable to more common techniques and potentially more advantageous.

Bibliography

[1] J. Aguilar, T. Zhang, F. Qian, M. Kingsbury, B. McInroe, N. Mazouchova, C. Li, R. D. Maladen, C. Gong, M. J. Travers, R. L. Hatton, H. Choset, P. B. Umbanhowar, and D. I. Goldman. A review on locomotion robophysics: the study of movement at the intersection of robotics, soft matter and dynamical systems. *CoRR*, abs/1602.04712, 2016.

[2] N. Akinci, G. Akinci, and M. Teschner. Versatile surface tension and adhesion for sph fluids. *ACM Trans. Graph.*, 32(6):182:1–182:8, Nov. 2013.

[3] L. Barnes, M. Fields, and K. Valavanis. Unmanned ground vehicle swarm formation control using potential fields. In *Control Automation, 2007. MED '07. Mediterranean Conference* on, pages 1–8, June 2007.

[4] L. E. Barnes. *A potential field based formation control methodology for robot swarms*. PhD thesis, University of South Florida, 2008.

[5] M. Dorigo, D. Floreano, L. M. Gambardella, F. Mondada, S. Nolfi, T. Baaboura, M. Birattari, M. Bonani, M. Brambilla, A. Brutschy, D. Burnier, A. Campo, A. L. Christensen, A. Decugniere, G. D. Caro, F. Ducatelle, E. Ferrante, A. Forster, J. M. Gonzales, J. Guzzi, V. Longchamp, S. Magnenat, N. Mathews, M. M. de Oca, R. O'Grady, C. Pinciroli, G. Pini, P. Retornaz, J. Roberts, V. Sperati, T. Stirling, A. Stranieri, T. Stutzle, V. Trianni, E. Tuci, A. E. Turgut, and F. Vaussard. Swarmanoid: A novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics Automation Magazine*, 20(4):60–71, Dec 2013.

[6] M. Duarte, J. Gomes, V. Costa, S. M. Oliveira, and A. L. Christensen. *Hybrid Control for a Real Swarm Robotics System in an Intruder Detection Task*, pages 213–230. Springer International Publishing, Cham, 2016.

[7] G. C. E. S. E. Simetti, A. Turetta and M. Cresta. Protecting assets within a civilian harbour through the use of a team of usvs: Interception of possible menaces. In *OCEANS*, 2010.

[8] S. W. Ekanayake. *Formation of networked mobile robots*. PhD thesis, Deakin Univeristy, 2009.

[9] A. Franchi, P. Stegagno, and G. Oriolo. Decentralized multi-robot encirclement of a 3d target with guaranteed collision avoidance. *Auton. Robots*, 40(2):245–265, Feb. 2016.

[10] J. R. J. M. J.A. Sauter, R.S. Matthews and S. Riddle. Swarming unmanned air and ground systems for surveillance and base protection. In *Proceedings of AIAA Infotect@Aerospace 2009 Conference*, Apr 2009.

[11] G. K. F. Kevin M. Lieberman and D. P. Garg. Decentralized control of multi-agent escort formation via morse potential function. In *ASME 2012 5th Annual Dynamic Systems and Control Conference joint with the JSME 2012 11th Motion and Vibration Conference*, pages 317–324, Oct 2012.

[12] P. LaRocque. Autonomous robot patrolling of a sparsely populated unknown environment. Master's thesis, Rochester Institute of Technology, Nov 2010.

[13] D. Laskowski. Asset protection in a limited swarm environment utilizing artificial potential fields. Master's thesis, Rochester Institute of Technology, Feb 2010.

[14] D.-W. Lee and K.-B. Sim. Artificial immune network-based cooperative control in collective autonomous mobile robots. In *Robot and Human Communication, 1997. RO-MAN '97. Proceedings., 6th IEEE International Workshop*, on pages 58–63, Sep 1997.

[15] H.-J. Lee and K.-B. Sim. Distributed moving algorithm of swarm robots to enclose an invader. In *Control, Automation and Systems, 2008. ICCAS 2008. International Conference* on, pages 729–733, Oct 2008.

[16] I. Mas, S. Li, J. Acain, and C. Kitts. Entrapment/escorting and patrolling missions in multi-robot cluster space control. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5855–5861, Oct 2009.

[17] A. R. Merheb, V. Gazi, and N. Sezer-Uzol. Implementation studies of robot swarm navigation using potential functions and panel methods. *IEEE/ASME Transactions on Mechatronics*, 21(5):2556–2567, Oct 2016.

[18] L. C. A. Pimenta, G. A. S. Pereira, N. Michael, R. C. Mesquita, M. M. Bosque, L. Chaimowicz, and V. Kumar. Swarm coordination based on smoothed particle hydrodynamics technique. *IEEE Transactions on Robotics*, 29(2):383–399, April 2013.

[19] J. L. S.Kazadi and J. Lee. Artificial physics, swarm engineering, and the hamiltonian method. In *World Congress on Engineering and Computer Science 2007*, San Francisco, CA, pages 623–632, Oct 2007.

[20] W. M. Spears and D. F. Gordon. Using artificial physics to control agents. In *Information Intelligence and Systems, 1999. Proceedings. 1999 International Conference* on, pages 281–288, 1999.

[21] L. Xie, J. Zeng, and Z. Cui. General framework of artificial physics optimization algorithm. In *Nature Biologically Inspired Computing, 2009. NaBIC 2009. World Congress* on, pages 1321–1326, Dec 2009.

Appendix A

The following is an instructional guide for setting up and running customized simulations using ARGoS and the control algorithms developed in this work in a Linux environment. It is assumed that ARGoS is already installed onto the Linux system. In addition, while this guide will cover the basics of setting up an ARGoS configuration file, it will not cover in-depth topics such as actuator and sensor parameters, physics engines, and visualizations among other topics. The basics of how to use the software libraries developed during this research will only be covered. ARGoS provides many examples and tutorials to learn more about the simulation environment itself.

User's Guide:

The ARGoS configuration file is a XML document using xml version 1.0 with **<argos-configuration>** as the root element. The children include **<framework>**, **<controllers>**, **<arena>**, **<physics_engines>**, **<media>**, and **<visualization>**. However, the most important elements for the simulation is the framework, controllers, and arena elements. An example configuration file will be shown at the end of the appendix.

Framework:

The framework element holds parameters relevant to how the computer runs the simulation itself and only involves a small number of parameters relative to the other elements. Below is an example of the framework element in which the computer will use a single thread to do all calculations. The experiment length will stop the simulation after the time specified (given in seconds). If the time is 0, then the experiment will continue until a user manually stops it with the user interface. The ticks_per_second parameter specifies how many iterations of the control algorithm's main loop should run every second. The random_seed parameter is used in cases where random numbers need to be consistent among multiple experiments.

```
<framework>
        <system threads="1" />
        <experiment length="0" ticks_per_second="10" random_seed="101" />
</framework>
```

Controllers:

There are three different elements that may be children of the **<controllers>** element: asset_controller, threat_controller, and agent_controller. Below is an example of the asset_controller child. All three types of children will require an unique id along with a path to the library to use. This path assumes that working directory is the root folder containing the "build" directory. For the asset, the path is "build/controllers/asset/ libasset.so." For the threats and agents, the paths are "build/controllers/threats/ libthreat.so" and "build/controllers/agents/libagent.so" respectively.

The actuators and sensor elements needed by all controllers are the same, and as such, not much detail will be discussed. All that's needed for the actuators is the differential steering, leds, and the range and bearing (which is the infrared communication transmitter). For the sensors, the proximity sensors are needed, along with the positioning sensor, and the range and bearing sensor (the infrared communication receiver).

The **<params>** element is where the customization of the controllers happens. The asset and threat controllers have the same general parameters, specifying 'x1' and 'y1' - the global point to which the robot should go first - and 'x2' and 'y2' - the second destination. The 'start' parameter specifies after how many seconds from the start of the simulation the robot should start moving, and the 'mode' parameter specifies what type of movement the robot should do. There are three options for the mode. The first is "s" where the robot is static and does not move at all. The second mode is "m" where the robot will move between the points specified by x1, y1, x2, and y2 indefinitely. Finally, the last mode is "b" where the robot will go to the first destination, to the second destination, and then stop permanently.

The **<params>** element for the agent's controller only has one parameter: mode. With this parameter, one can specify if the SPH based algorithm should be used by setting mode equal to "s". If mode is not set to "s", then it must be set to "p" which will mean that the swarm of agents will use the potential fields based method.

All three controllers have a sub-element of **<wheel_turning>** which specifies certain angles in radians to serve as the threshold between a hard turn and a soft turn, and a soft turn and no turn. A hard turn is one in which the robot stops and spins while a soft

turn is one in which the robot spins in a particular direction but does so while moving. The last parameter is the maximum speed in centimeters per second.

```
<asset_controller id="asset" library="build/controllers/asset/libasset.so">
        <actuators>
                <differential_steering implementation="default" />
                <leds implementation="default" medium="leds"/>
                <range_and_bearing implementation="default"/>
        </actuators>
        <sensors>
                <footbot_proximity implementation="default" />
                <positioning implementation="default" />
                <range_and_bearing implementation="medium" medium="rab"/>
        </sensors>
        <params x1="5" y1="0" x2="-10" y2="6" start="20" mode="b">
                <wheel_turning hard="1.5708" none="0.05" max="5" />
        </params>
</asset_controller>
```

Arena:

The **<arena>** element is where one can specify the environment of the simulation. The parameters of this element specify the size and center of the arena. To put objects into the arena such as walls, one can add children elements as shown in the example configuration file at the end of this User's Guide.

One such children element would be a robot model which in this case would be a footbot. To add a single robot to the environment, one can add the **<foot-bot>** element with a unique id. Under this, two more elements should be added, the first of which is the body. This will specify the position and orientation of the robot while the second element specifies which controller the robot should use.

```
<foot-bot id="asset">
        <body position="15,10,0" orientation="0,0,0" />
        <controller config="asset"/>
</foot-bot>
```

However, if the end goal is to add many robots in a confined space placed randomly, the **<distribute>** element should be added as a child to the arena. This includes a position tag which allows one to specify the lower left corner and the upper right corner of the square area in which the robots should be distributed. The orientation tag specifies how the orientation of robots should be decided while the entity tag specifies items like the number of robots and the controller each of the robots should use. Below is an example of one hundred robots scattered randomly around a four square meter area.

```
<distribute>
        <position method="uniform" min="13,8,0" max="17,12,0"/>
        <orientation method="gaussian" mean="0,0,0" std_dev="360,0,0"/>
        <entity quantity="100" max_trials="100">
                <foot-bot id="agent">
                        <controller config="agent"/>
                </foot-bot>
        </entity>
</distribute>
```

To find out more about certain parameters and elements not covered in this guide, visit the ARGoS website. The following is an example of a complete configuration file.

```
<?xml version="1.0" ?>
<argos-configuration>

    <framework>
        <system threads="1" />
        <experiment length="0" ticks_per_second="10" random_seed="101" />
    </framework>

    <controllers>
        <asset_controller id="asset" library="build/controllers/asset/libasset.so">
            <actuators>
                <differential_steering implementation="default" />
                <leds implementation="default" medium="leds"/>
                <range_and_bearing implementation="default"/>
            </actuators>
            <sensors>
                <footbot_proximity implementation="default" />
                <positioning implementation="default" />
                <range_and_bearing implementation="medium" medium="rab"/>
            </sensors>
            <params x1="5" y1="0" x2="-10" y2="6" start="20" mode="b">
                <wheel_turning hard="1.5708" none="0.05" max="5" />
            </params>
        </asset_controller>

        <threat_controller id="threat" library="build/controllers/threat/libthreat.so">
            <actuators>
                <differential_steering implementation="default" />
                <leds implementation="default" medium="leds"/>
                <range_and_bearing implementation="default"/>
            </actuators>
            <sensors>
                <footbot_proximity implementation="default" />
                <positioning implementation="default" />
                <range_and_bearing implementation="medium" medium="rab"/>
            </sensors>
            <params x1="0" y1="5" x2="10" y2="5" start="0" mode="m">
                <wheel_turning hard="1.5708" none="0.05" max="10" />
            </params>
        </threat_controller>
```

```
    <agent_controller id="agent" library="build/controllers/agent/libagent.so">
        <actuators>
            <differential_steering implementation="default" />
            <leds implementation="default" medium="leds"/>
            <range_and_bearing implementation="default"/>
        </actuators>
        <sensors>
            <footbot_proximity implementation="default" />
            <positioning implementation="default" />
            <range_and_bearing implementation="medium" medium="rab"/>
        </sensors>
        <params mode="p">
            <wheel_turning hard="1.5708" none="0.05" max="50" />
        </params>
    </agent_controller>

</controllers>


<arena size="41, 41, 1" center="0,0,0.5">

    <box id="wall_north" size="40,0.1,0.5" movable="false">
        <body position="0,20,0" orientation="0,0,0" />
    </box>

    <box id="wall_south" size="40,0.1,0.5" movable="false">
        <body position="0,-20,0" orientation="0,0,0" />
    </box>

    <box id="wall_east" size="0.1,40,0.5" movable="false">
        <body position="20,0,0" orientation="0,0,0" />
    </box>

    <box id="wall_west" size="0.1,40,0.5" movable="false">
        <body position="-20,0,0" orientation="0,0,0" />
    </box>


    <foot-bot id="asset">
        <body position="15,10,0" orientation="0,0,0" />
        <controller config="asset"/>
    </foot-bot>
```

```
        <foot-bot id="threat">
            <body position="7.5,5,0" orientation="0,0,0" />
            <controller config="threat"/>
        </foot-bot>

        <distribute>
            <position method="uniform" min="13,8,0" max="17,12,0"/>
            <orientation method="gaussian" mean="0,0,0" std_dev="360,0,0"/>
            <entity quantity="100" max_trials="100">
                <foot-bot id="agent">
                    <controller config="agent"/>
                </foot-bot>
            </entity>
        </distribute>
    </arena>

    <physics_engines>
        <dynamics2d id="dyn2d" />
    </physics_engines>

    <media>
        <led id="leds" />
        <range_and_bearing id="rab"/>
    </media>

    <visualization>
        <qt-opengl />
    </visualization>

</argos-configuration>
```