# ABSTRACT

Title of thesis: OVERCOMING LOCAL MINIMA THROUGH VISCOELASTIC FLUID-INSPIRED SWARM BEHAVIOR

Loy McGuire, Master of Science, 2020

Thesis directed by: Dr. Michael W. Otte
Department of Aerospace Engineering

My paper discusses a novel swarm robotic algorithm inspired by the open channel siphon phenomena displayed in certain viscoelastic fluids. This siphoning ability enables the algorithm to mitigate the trapping effects of local minima, which are known to affect physicomimetics-based potential field control methods. Once a robot senses the goal, local communication between robots is used to propagate path-to-goal gradient information through the swarm's communication graph. This information is used to augment each agent's local potential field, reducing the local minima trap and often eliminating it. In this paper real world experiments using the Georgia Tech Miniature Autonomous Blimp (GT-MAB) aerial robotic platforms as well as mass Monte Carlo test simulations conducted in the Simulating Collaborative Robots in Massive Multi-Agent Game Execution (SCRIMMAGE) simulator are presented. Comparisons between the resultant behaviors and potential field based swarm behaviors that both do, and do not incorporate local minima fixes were assessed. These experiments and simulations demonstrate that this method is

an effective solution to susceptibility to local minima for potential field approaches for controlling swarms.

# OVERCOMING LOCAL MINIMA THROUGH VISCOELASTIC FLUID-INSPIRED SWARM BEHAVIOR

by

Loy McGuire

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2020

Advisory Committee:
Dr. Michael W. Otte, Chair/Advisor
Dr. Huan Xu
Dr. David Akin

# Dedication

I would like to dedicate this work to my family who have given me more support and encouragement than I could have ever asked for.

# Acknowledgments

I would like to share my gratitude towards my advisor, Dr. Michael Otte, for giving me the opportunity to explore my passions further. I appreciate the care you have shown for the academic and personal well-being of myself and all of your students, and the guidance you have given me through my graduate student experience so far. I would also like to thank Dr. David Akin and Dr. Mumu Xu for granting me their time, patience, and advice as members of my committee.

To the institutions of the Naval Research Laboratories, the University of Maryland, and Miami University, I extend my thanks and appreciation for providing me with mentors, peers, and experiences that have fostered the person I have become today.

# Contents

# List of Figures

# List of Abbreviations

| | |
|---|---|
| GT-MAB | Georgia Tech-Miniature Autonomous Blimp |
| IR | Infrared |
| LJP | Lennard-Jones Potential |
| ROS | Robot Operating System |
| SCRIMMAGE | Simulating Collaborative Robots in Massive Multi-Agent Game Execution |

# List of Algorithms

# Chapter 1

## Introduction

The field of path planning is broad and the application of appropriate methods for certain scenarios can be complex and nuanced. Path planning can be broken down depending on the type of system that requires it. One way is by organizing it into single-agent, multi-agent, and swarm systems. Single-agent systems are the simplest, only needing to focus on it's own movement. Multi-agent systems add complexity since agents need to be coordinated and avoid collisions. They may have to use communication networks to share information between themselves or with a central controller. The application of swarm systems takes this further in using mass numbers of agents to accomplish coordinated goals.

The benefit of swarms is revealed by it's distinction from multi-agent systems. While both types of systems use multiple agents, swarms have the property of scalability with respect to the number of agents within it's system. Multi-agent planning causes factors such as computation to increase exponentially as agents are added to the system. Swarms are able to increase their numbers without causing the system to fail.

The ability to scale opens many avenues to contribute positive capabilities to the system, many of which take inspiration from and are displayed by biological swarms. Ants work together to collectively lift objects that are too large for an

individual to carry and have been observed to build bridges by using ants in the swarm to hang onto each other until their bodies fill in a gap. Bees also display swarming tendencies through using large numbers to fend off predators disturbing their nests. The ability to scale their numbers also benefits swarms by adding redundancy measures, allowing swarm systems to accomplish objectives even when agents within the system experience failures.

## 1.1 Motivation

To develop swarm capabilities for robots, the methods used to control them must allow swarms to scale in size. This requires the swarm system's control method to handle increasing computational costs and communication complexity. To reduce the complexity of path-finding computation and communication systems, and prevent saturating communication bandwidths, different methods can be implemented to direct the movement of swarms. One such method is by using potential fields, which were originally used by Khatib [1] for controlling robot manipulator arms but translate well for swarm movement. In potential fields methods all positions within a workspace are associated with a magnitude. The gradient of the potential field can be interpreted as a vector field imposed on the workspace and used to control an agent. Usually, all agents are similarly affected by the vector field. This translates well for swarms since individual agents can compute the potential field local to them with little to no communication required, allowing the computation to be distributed throughout the robots in the system. Therefore as more agents

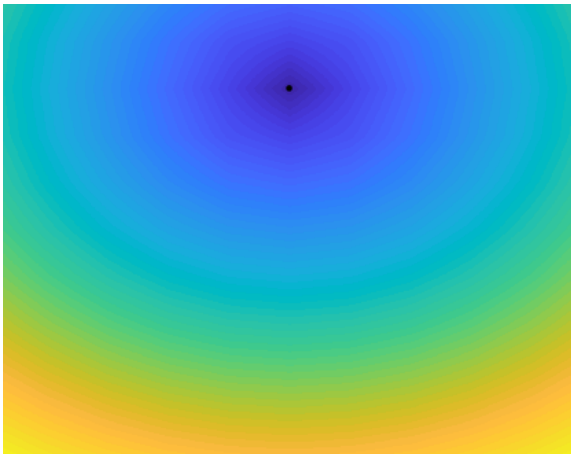
are added to the system, more computational power is added as well.

One weakness of potential fields occurs when an area has a local minimum potential surrounded by higher potentials, as seen in Figure 1.1c. This causes the vector field to point inward, trapping agents in these areas and preventing them from reaching the global minimum potential. This phenomenon is known as a local minima trap. This thesis discusses a novel approach to combat the negative effects local minima traps have when using potential fields methods to control a swarm.

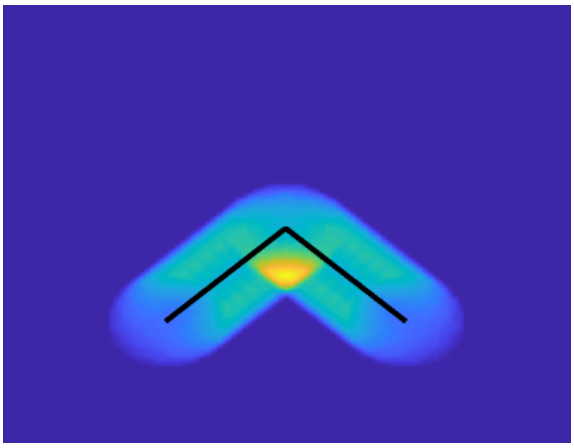
## 1.2 Chain Siphon Method

My work introduces the chain siphon algorithm as a method of overcoming the local minima trap. The chain siphon method leverages the inherent size of a swarm to manipulate the local potentials of each agent to navigate them out of local minima traps. In order to do this, it is assumed that the number of swarm agents is large enough to create a chain of local communications from the goal area to the local minimum trap. Through this communication chain, as seen in Figure 1.2, the agents form a queue starting with the agents sensing the goal. The rest of the swarm follows the agents ahead of them in the queue until they are all led to the goal area.

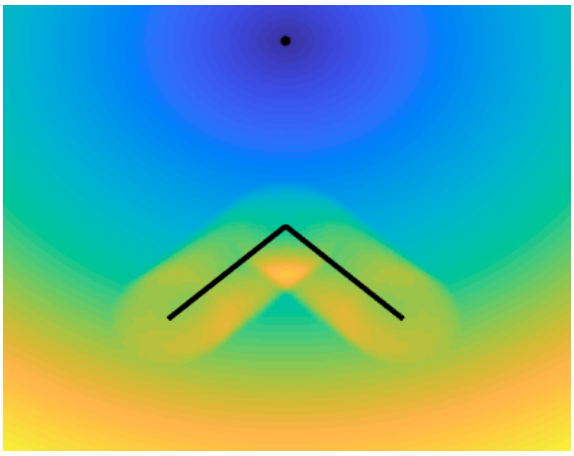
The emergent behavior of the swarm simulates a viscoelastic fluid exhibiting the open channel siphon effect. Fluids characterized by this phenomenon are able to "self-pour"; once the material starts pouring out of a container, it is able to pull more of the material up and over the container walls. For swarm movement this allows the agents to move around obstacles like the particles of a liquid while also



(a) Modeling a goal position as the global minimum potential.



(b) Obstacle modeled as a high potential area.



(c) Summation of the two potential maps.

Figure 1.1: An agent following the gradient of potential fields would move towards the goal in (a) and be repelled from the obstacle in (b). This causes the low potential area surrounded by higher potentials in (c), a local minimum trap.

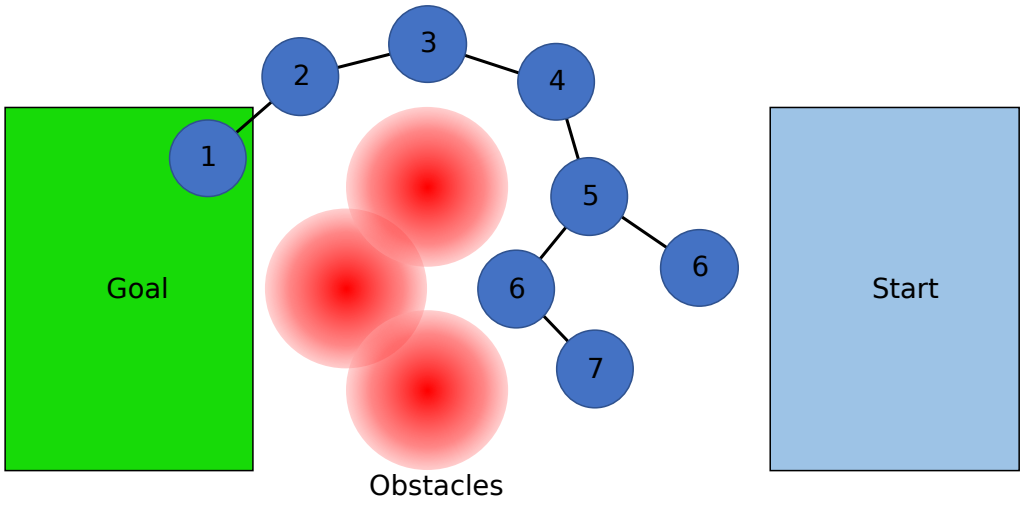



Figure 1.2: A visualization of the local chain of communication applied during chain siphon behavior. The queue begins with the agent arriving within sensing range of the goal and propagates backwards to neighboring agents. Note that the agent with a queue value of 7 establishes a link with the closest agent with a queue value of 6.

enabling them to pull themselves out of local minima traps.

The chain siphon algorithm is tested in simulations and with real world robotic experiments using a swarm of the Georgia Tech-Miniature Autonomous Blimps (GT-MABs) in order to examine it's performance. While the hardware testbed experiments are set up as a centralized system, they are designed to be easily transitioned into a distributed system with proper hardware. The central ground station separately communicates and controls all of the blimps, which in turn move within the workspace. Sensor and communication restraints are artificially implemented within the ground station. Despite these centralized factors the behavior of the algorithm translates well to a swarm system because all sensing and communication can be done at a local level. Within this work's implementation only the goal seeking behavior requires global knowledge of the goal position, although this could be

modified to fit the needs of different systems.

## 1.3 Related Works

The work discussed can relate to several different fields of research. Section 1.3.1 explores different modes of controlling swarm systems. A common approach using potential fields in multi-agent and swarm settings is to coordinate formation control, discussed in Section 1.3.2, and to implement different behaviors into the control function, discussed in Section 1.3.3. Other methods of navigating local minima traps are given in Section 1.3.4. Original works and different use cases for physicomimetics are presented in Section 1.3.5, and the development of the GT-MABs are in Section 1.3.6.

### 1.3.1 Swarm Motion

Many different methods are available to control swarms [2]. Although distributed systems are helpful for swarms, centralized systems can be used as well. Becker et al. [3] use a common input to control mass numbers of simple robots by manipulating them with the environment. This type of movement has been shown to utilize complex environments [4] and systems with agent-unique dynamics [5] to benefit the system. Another approach to swarm control is to model and direct them as a traffic problem [6] [7]. Systems modeled like this can use Quadratic Unconstrained Binary Optimization (QUBO) to optimize movement between multiple swarms [8]. These methods differ from ours by using a centralized systems and tak-

ing a high level control approach, while the chain siphon method uses distributed control for individual agents to produce emergent behavior.

### 1.3.2 Potential Field Formation Control

One approach to swarm control is by using specific functions to keep formation control while trajectories are generated for the swarm system to follow using single agent path planning methods such as Dijkstra's [9], A* [10], RRT [11], and PRM [12]. Barnes et al. has used bivariate normal functions [13] or sigmoid and normal functions [14] to construct formation control equations. Agents have also been modeled as spring-damper systems [15] to keep spacecraft swarms in formation. Bentes and Saotome [16] developed trajectories using A* [10] that the swarm center would localize around and follow. Other work [17] has used sampling based path planning to create the swarm trajectory and used potential fields to drive the swarm along it applying stochastic reachable sets as repulsive forces to dynamic obstacles. The chain siphon method does not require the swarm as a whole to follow a trajectory, instead each agent acting individually and interacting with it's surroundings causes an emergent behavior to form across the swarm.

### 1.3.3 Potential Field Control Behaviors

Potential fields are useful when implementing different control behaviors, which can be weighted using Particle Swarm Optimization [18] to optimize which behaviors are most prevalent [19]. This method could be used on the chain siphon method

but differs by modifying behavior weights rather than implementing a new behavior. Control behaviors have been displayed [20] with formation control using virtual structures as well as collision avoidance using social potential fields between agents [21]. The chain siphon method is not a collision avoidance method but can be used in addition to them within the control equation. Real and virtual agents have also been used as leaders for swarm control [22] by trying to match the swarm's velocity and headings with the leader's. The chain siphon method's use of a queue could be said to make some agents leaders, but this is a function of what they sense and is not a designation given a priori.

### 1.3.4 Local Minimum Trap Avoidance

There are multiple methods that have been used to avoid the local minima traps for multi-agent movement. By using dynamic vector field generation [13][14] many of the situations that cause agents to be trapped can be avoided. The agents using chain siphon behavior do not passively rely on dynamic field generation but use the method to actively avoid traps. By using dynamic internal fields [23] agents are able to recognize the time interval they are in the local minima and after enough time they force those areas to have higher potentials so they can escape. The method in this paper does not rely on time intervals to initialize it's behavior. Similarly, using vortex inspired behavior and Brownian motion [24] agents are able to escape local minima by increasing the energy they attribute to Brownian motion when they sense they are stuck. Stochastic lateral disturbances have been used to avoid traps

[25] [26], and in the case of agents causing local minima for another or multiple other agents then agents are assigned priorities and their speeds are augmented to avoid interfering with other agents' trajectories [27]. The chain siphon method does not rely on stochastic movement in order to avoid local minima.

### 1.3.5 Physicomimetics

The DAEDALUS work from Hettiarachchi and Spears [28] uses an algorithm dictated by the Lennard-Jones potential (LJP) equation [29] as the equation to determine collision avoidance and swarm cohesion. The control equation in my work uses this in addition to the goal seeking, obstacle avoidance, and chain siphon behaviors. The LJP model is similar to a Boids [30] model of swarm movement, although the alignment factor is moot as the agents, both GT-MABs and simulated, move holonomically. This is categorized in the field of physicomimetics [31], a section of potential fields methods [1] mimicking swarm behavior from physical phenomena. I have used these previous works to add behavior that reduces the cost of running into local minima in certain scenarios.

Physicomimetics has found use in many different scenarios. Sydney et al. [32] used a Bayesian target detection method to modify a temperature gradient map that augmented the the speeds of agents within a space. Systems have simulated pheromone dispersion as a stigmergetic information indicator as an aid for search algorithms [33]. Modeling agents as particles within a gas has also shown benefits to agents as a coverage and obstacle avoidance method [34]. While these have different

applications than the chain siphon method, the approaches contain similarities.

The Lennard-Jones potential driven model has contains a likeness to the model from Tanner et al. on flocking [35][36] and multi-robot movement [37]. This work modeled a Boids behavior with separation, alignment, and cohesion elements that were based off of a potential function, albeit not the Lennard-Jones potential function.

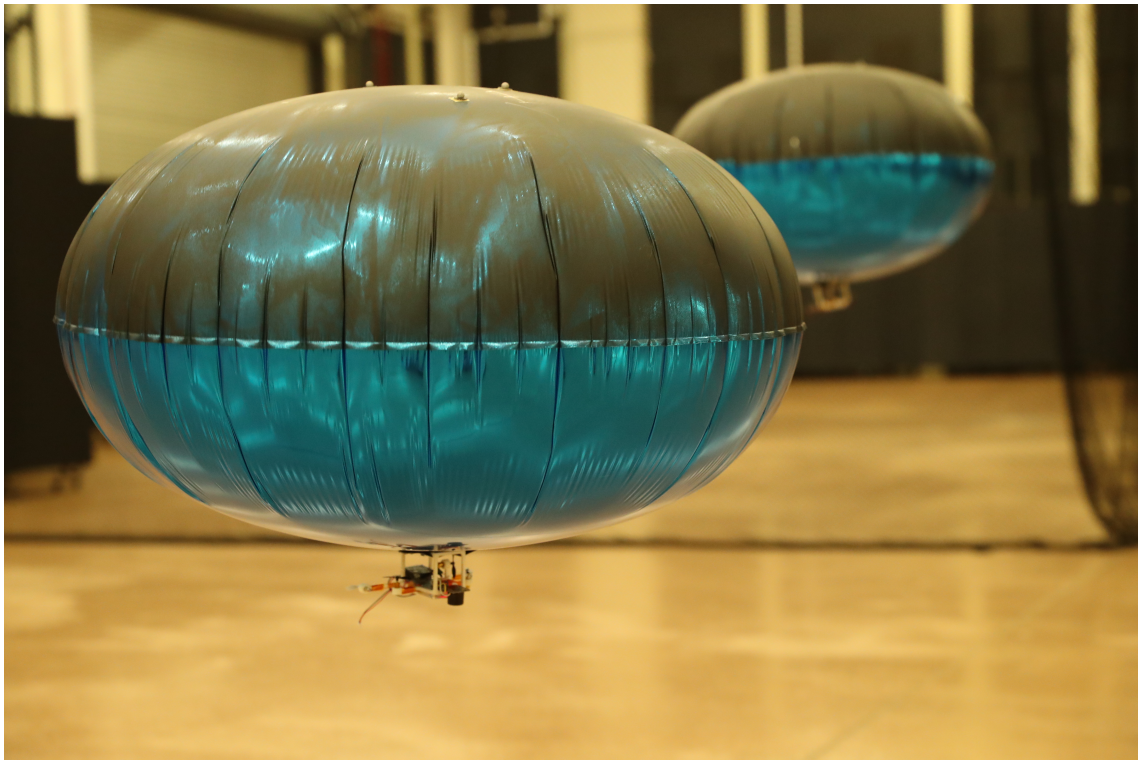


Figure 1.3: A close up picture of the GT-MAB hardware testbed used in the experiments.

### 1.3.6 GT-MABs

The GT-MABs, as seen in Fig 1.3, were used as the robot for the experiments discussed in this thesis. The dynamic model for the blimps was first presented in [38]

following procedures of previous works [39][40], and improved upon by considering pitching, rolling, and aerodynamic damping effects [41]. A control system is built around their models for actuating the GT-MABs for my experiments.

They have been used for a variety of applications. They have demonstrated their ability to be deployed as a swarm system [42]. By syncing their movements in an A* planned path [43], they demonstrated a time-based pattern control. They were also used to test gesture controls allowing a single person to send commands to a multi-agent system [44].

## 1.4 Contributions of Thesis

This work contributes to the field of swarm movement, but more specifically to the field of swarm movement using potential fields to navigate environments. Works using potential fields generally discuss the drawbacks attributed to the local minima traps, although potential fields methods within swarm systems can be powerful tools due to their ease of implementation into a distributed system. This work provides another type of method which aids the field of swarm systems in accessing the application of potential fields methods while reducing the drawbacks.

The solutions discussed earlier depend on stochastic methods that rely on their randomness to escape local minima traps, usually increasing the magnitude of the stochastic movements over time until their agents reach areas that will not lead them back into the traps. This work contributes the use of more directed efforts that don't rely on stochastic processes, allowing more efficient movement that could

save time and fuel.

It also adds to the area of distributed systems, a hurdle keeping a number of multi-agent methods from being available for swarm systems. Many multi-agent methods that are not distributed break down as the number of agents in the system increases, often due to limitations in computation or communications. Developing the method to be available for distributed systems use allows for more capabilities with swarm systems.

## 1.5 Outline of Thesis

Chapter 2 goes into the methodology of the swarm behaviors involved. In Section 2.1 the technical approach of the chain siphon method is laid out and the algorithm is displayed and discussed in detail. In Section 2.2 the assumptions made for this work are discussed. Section 2.3 details all the behaviors that govern the potential fields of each agent, and provides the full control equation. The equations used for goal seeking and obstacle repulsion, as well as the algorithm to determine how obstacles influence an agent's local potential field can be found in in Section 2.4. The derivation of a velocity term for the Lennard-Jones potential and the algorithm it is incorporated in are described in Section 2.5, as well as the method of using a leader heuristic. Section 2.6 discusses the initial technical approaches looked at that ultimately evolved into the chain siphon method. Section 2.6.1 was an attempt using a different natural phenomenon similar to the open channel siphon effect. A discussion of the need for dynamic linking is in Section 2.6.2 which gives examples

of early issues found when using a a static queue generation method. The issues associated with not limiting the maximum queue value is gone into in Section 2.6.3.

The preparation and guidelines for experiments and simulations conducted to gather data on the different behaviors makes up the content in Chapter 3. Section 3.1 and 3.2 summarize how the simulations and robotic experiments set up their environments and run. The implications of using the blimp platform in the physical world is shared in Section 3.3. Section 3.4 discusses how parameters were selected and which parameters were used in the simulations and hardware experiments.

Chapter 4 provides the results of the simulations and experiments set up in the previous chapter. The results of the 2-D and 3-D simulations in Section 4.1 shows the data and the analysis of that data. The same type of content is given for the hardware experiments in Section 4.2. The variation in the number of robotic agents is discussed here as well.

The paper is concluded in Chapter 5 where a discussion of the simulation results can be found in Section 5.1 and the discussion of hardware experiment results is in Section 5.2. Section 5.3 ends the paper by examining how my work can be expanded upon in future works.

# Chapter 2

# Algorithm Development And Integration Into A Boids-Like Method

In order to evaluate the performance of the chain siphon method (Section 2.1), a full control equation (Section 2.3) was developed to direct the swarm towards a goal and avoid obstacles (Section 2.4), which creates the local minimum trap, and gives the swarm a holonomic boids-like behavior with the Lennard-Jones potential method and creates a structure to implement a leader heuristic behavior(Section 2.5). The development of the chain siphon method evolved from previous methods (Section 2.6) which had failure points prompting improved iterations of the algorithm.

## 2.1 Physicomimetic Open Channel Siphon Effect

The chain siphon method comes into effect when an agent senses the goal. Each agent that senses the goal begins a queue which they are at the front of and they broadcast their position in the queue so that neighboring agents can know. These neighboring agents then place themselves in the next position in the queue and broadcast their queue placement to their neighbors. This process repeats until there are no more neighboring agents within range of the queued agents. The robots then follow their neighbors that are ahead of them in the queue until they all can sense the goal.

In algorithm 1 line 1 assigns the total number of agents in the swarm $n_s$ as

**Algorithm 1** Siphon Behavior

$n_s \leftarrow$ Total number of agents in the swarm
**loop**
    $q = n_s$ ▷ Default position in queue
    $senseGoal()$
    **if** goal is found **then**
        $q = 1$
    $q_{min} = n_s$
    $d_{min} \leftarrow$ Maximum neighbor range
    $queryNeighbors()$
    **for** every neighbor $x$ **do**
        $v_x = x.pos - self.pos$ ▷ Rel. position vector
        $d = \|v_x\|$
        **if** $x.q < q_{min}$ **then**
            $q_{min} = x.q$
            $d_{min} = d$
            $v_{link} = v_x$
        **if** $x.q = q_{min}$ **and** $d < d_{min}$ **then**
            $d_{min} = d$
            $v_{link} = v_x$
    **if** $q_{min} < q$ **then**
        $q = q_{min} + 1$
    **if** $q > n_s$ **then**
        $q = n_s$
    $broadcast(self.ID,\ q)$
    $C_{siphon} = v_{link}.normalize()$
    **return** $C_{siphon}$

the maximum possible queue position. The queue value $q$ of the agent is assigned $n_s$ as it's default value in line 3. If the agent is within sensor distance of the goal then it changes $q$ to a value of 1 in line 6.

The default minimum queue value $q_{min}$ of the neighboring agents $x$ and the minimum distance value $d_{min}$ are assigned their maximum possible values in lines 7 and 8 respectively. The agent searches for neighboring agents and their associated queue values $x.q$ and positions $x.pos$ then finds their relative position $v_x$ in line 11 and distance $d$ in line 12. The current $q_{min}$ is compared with $x.q$, and if $x.q$ is found to be lower then it's value is placed as the new $q_{min}$, it's $d$ as $d_{min}$, and $v_x$ is designated as the vector pointing to the agent that is to be followed $v_{link}$ in lines 14, 15, and 16. If $x.q$ is the same value as the current $q_{min}$ then the agent checks if the neighbor is closer than the previous $d_{min}$. If it is $d_{min}$ is given the new closest value and $v_{link}$ is set so it follows the neighbor in lines 18 and 19. This allows the agent to follow the neighbor with the lowest $x.q$ that it is closest to.

Once the $q_{min}$ between all neighbors has been identified, the agent compares it to its own $q$. If $q_{min}$ is found to be less than $q$ then in line 21 $q$ is given a value of 1 more than $q_{min}$, except when it is greater than the maximum queue number $n_s$.

Once the final value for $q$ is determined, the agent broadcasts a message containing its ID with $q$ to the neighboring agents in line 24. Finally in line 25 $v_{link}$ is normalized and the resulting vector is assigned to $C_{siphon}$.

Weighting $C_{siphon}$ sufficiently more than the other portions of the control equation causes the agent to follow the neighbor it has linked onto. Only control factors that are able to approach infinity will have a noticeable effect once the agent is

linked, such as the obstacle or collision avoidance terms. This causes other behaviors such as adhesion between agents and goal seeking to become negligible.

## 2.2 Assumptions

The chain siphon effect relies on several assumptions and generalisations in order to showcase it's effectiveness against local minima traps. The first is that there are enough agents in the swarm to be able to sense a goal and link to agents stuck in local minima using a local chain of communication. Another assumption is that the $queryNeighobors()$ function in Algorithm 1 has been developed such that agents will not try to link through obstacles to other agents. This can be equivocated with the ability to both sense and communicate with other agents, such as an agent needing both line-of-sight for their sensors to detect an agent and requiring them to be within communications range. Within my simulations and experiments the distance that agents were repelled by obstacles between them was larger than the distance they would associate with an agent as a neighbor.

The other function in Algorithm 1 that has been generalized is how the goal is sensed in the $senseGoal()$ function. This requires that an agent is able to sense a goal, whether it is a temporary goal or a global minimum, and it is able to reach the goal. Within my implementation, an agent senses a goal once it reaches a certain distance that allows it to be within sensor range. This works in the scenario set up in the simulations and experiments, but may be leading agents into a new local minimum if an agent senses the goal but obstacles obstruct the agent from reaching

it.

## 2.3 Control Equation

For showcasing the effect of chain open channel siphon, the algorithm was incorporated into a swarm being directed by goal seeking, obstacle avoiding, and flocking behaviors. The flocking behavior chosen for our swarm used a derivative of the Lennard-Jones potential similar to Hettiarachchi et al. [28] for determining our base behavior. The full control equation is as follows:

$$\begin{aligned} \mathbf{V} = \mathbf{W_{goal}} * \mathbf{C_{goal}} + \mathbf{W_{obstacle}} * \mathbf{C_{obstacle}} \\ + \mathbf{W_{LJP}} * \mathbf{C_{LJP}} + \mathbf{W_{siphon}} * \mathbf{C_{siphon}} \end{aligned} \tag{2.1}$$

This equation is a velocity controller, denoted by the vector $\mathbf{V}$, which contains scalar weighting factors $\mathbf{W_i}$ and control vectors $\mathbf{C_i}$ for each component of their respective behaviors. The controller also limits the maximum speed.

An agent's potential fields in the workspace assign the goal position as a global minimum potential while obstacles are given potentials with magnitudes approaching infinity. Interactions between agents use the Lennard-Jones potential method which directs the swarm's flocking and viscoelastic fluid-like behaviors. The repulsion factor of the Lennard-Jones potential method acts similar to the incompressibility of the fluid, and the cohesion of the agents causes the viscous "flow" of the agents pulling each other when moving around obstacles.

Concave obstacle configurations can create a local minimum that may trap agents. As agents heading toward the goal they will tend to fill in the local minimum

trap until it is saturated, causing other agents to be repelled from the area due to collision avoidance from the agents that are stuck. While the Lennard-Jones potential behavior can be influential enough to destabilize some agents from a local minimum trap, it is likely that there will still be agents stuck after most of the swarm has passed by the obstacles. These scenarios are where the chain siphon method shows its greatest value, as it is able to use the structure of the swarm to propagate communication and sensor data backwards in order to free the agents so the entire swarm is able to reach the goal with a reduced loss of agents.

## 2.4 Goal Seeking and Obstacle Avoidance

The goal seeking vector is determined through the following equation:

$$\mathbf{C_{goal}} = \left( \frac{\mathbf{P_{goal}} - \mathbf{P_{current}}}{\|\mathbf{P_{goal}} - \mathbf{P_{current}}\|} \right) * v_0 \tag{2.2}$$

where $\mathbf{P_{current}}$ and $\mathbf{P_{goal}}$ are the positions of the agent and the goal, respectively. The resulting normalized vector is multiplied by a specified scalar initial velocity $v_0$ to give the vector a desired magnitude.

**Algorithm 2** Obstacle Repulsion

```
1: a ← Obstacle repulsion value
2: loop
3:     senseObstacles()
4:     for every sensed obstacle x do
5:         v_x = x.pos − self.pos
6:         d = ||v_x||
7:         v_x.normalize()
8:         r = −(v_x a)/d^2
9:         C_obstacle += r
10:    return C_obstacle
```

In algorithm 2 line 5, for each obstacle the agent finds the vector between the position of the agent and the obstacle $\mathbf{v_x}$. If the obstacle is within the sensor range of the agent, then the vector is normalized in line 7 and multiplied by the negative of the repulsion value $a$ and the inverse distance squared in line 8. This value is added to the final vector $\mathbf{C_{obstacle}}$ in line 9.

## 2.5 Lennard-Jones Potential And Using A Leader Heuristic

The Lennard-Jones potential acts similar to the boids model, where it exhibits cohesion and avoidance, although it creates the emergent behavior of forming lattice structures from the agents when in steady flight and acts similar to a viscoelastic fluid when the swarm moves around an obstacle. To perform this behavior, the equation is defined as:

$$LJP_r = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right], \tag{2.3}$$

where $LJP_r$ is the intermolecular potential, $\sigma$ is the distance at which $V$ is zero, $r$ is the distance from atom $i$ to atom $j$, and $\epsilon$ is a parameter specifying the strength of their interactions.

Hettiarachchi and Spears [31] used Equation 2.3 to find the force of the interaction between particles so they could use it to drive their swarm behavior by taking the negative of its derivative,

$$F = -\left(\frac{d(LJP_r)}{dr}\right), \tag{2.4}$$

which becomes

$$F = -4\epsilon\left[\frac{-12\sigma^{12}}{r^{13}} + \frac{-6\sigma^{6}}{r^{7}}\right], \tag{2.5}$$

Taking the derivative of this once more and generalizing it gives the velocities of the agents,

$$v = 24\epsilon\left[\frac{-26b\sigma^{12}}{r^{14}} + \frac{7c\sigma^{6}}{r^{8}}\right], \tag{2.6}$$

where $b$ and $c$ can be specified to change the attractive and repulsive movement of the agent.

**Algorithm 3** Lennard-Jones Potential Behavior

1: $\epsilon \leftarrow$ Magnitude of attraction
2: $b \leftarrow$ Generalized parameter
3: $c \leftarrow$ Generalized parameter
4: **loop**
5: $queryNeighbors()$
6: **for** every neighbor x **do**
7: $v_x = x.pos - self.pos$ ▷ Rel. position vector
8: $d = \|v_x\|$
9: $v_x.normalize()$
10: $r = 24\epsilon\left[\frac{-26b\sigma^{12}}{c^{14}} + \frac{7d\sigma^{6}}{d^{8}}\right]v_x$
11: $C_{LJP}$ += $r$
12: **return** $C_{LJP}$

Algorithm 3 is similar to the one used to compute $\mathbf{C_{obstacle}}$ with the Lennard-Jones derived velocity component used in line 10 instead of the inverse squared distance.

The system has been set up so that it can be applied to a distributed system. The only global information needed is the relative goal location, but this could be substituted with an initial direction of the goal. Obstacles can be sensed in the real world instead of having their information taken from a centralized matrix. The

Lennard-Jones potential method only requires an agent to sense its neighbors and know their distance and direction from them. The chain siphon method communicates locally between agents that it has sensed to be its neighbors.

Using a leader heuristic to take swarms out of local minima was used by Mabrouk et al. [23] to guide a swarm out of an obstacle in a box configuration with a hole big enough for agents to move through. The agents are initially using random motion, they have global communication, and they have a global knowledge of where the local minima is and where the area outside the local minima is. To implement this method in my scenario I have restricted communication to local agents, but I have designated areas outside of the local minima trap so agents know when to use the heuristic. In Algorithm 3 after line 10 an agent would sense if a neighbor is outside of the local minima. If they are, the agent increases the attraction to that neighbor by multiplying $r$ by a leader heuristic parameter $e$.

## 2.6 Other Attempts

The chain siphon method was developed after previous methods failed to produce desirable results. Each failure point that was discovered was analyzed to determine the root cause. Most analysis required examining all forces acting on an agent during the time of failure, as well as how they were linked to other agents. Once the causes of failure were isolated, solutions were implemented into the behavior.

### 2.6.1 Chain Fountain Effect

One attempted method was modeled after the chain fountain effect, or Mould effect. This is a phenomenon where a long chain of small balls sits within a container and has one end thrown out of the container with sufficient force. The reactive forces of the balls and the rigid links moving out of the container causes the balls they are connected with to follow them so more of the chain is pulled out. This continues until the process is disrupted or the entire chain is pulled from the container.

The method was developed by using the same local chain of communication method that the chain siphon effect implements, but the chain of communication is static and cannot be reconfigured. Instead of agents querying their neighbors to find agents furthest to the front of the queue, agents towards the front of the queue query their neighbors to see where they are able to link backwards. The closest neighbor to the querying agent was linked and given the next position in the queue. This would cause the newly linked agent to perform the same process to find a neighbor to link to, and the process would repeat until there are no more agents to link with.

This method breaks down once it reaches clusters of agents, such as the ones within the local minima traps. While the intention was that as the chain moved towards the goal and away from the trap the unlinked agents within the clustered group would file into the trap and link to the end of the chain, what was found to happen was the link would often intersect itself. This would cause an agent to attempt to move across where two agents linked with each other further down the chain. The agent attempting to move across the link would be repelled by

it's collision avoidance from the two other agents, preventing it from following the agent it was linked with. This would result in the agent attempting to follow it's linked agent after it had already moved past the obstacle, therefore running into the obstacle and remaining stuck. If the agent it was linked with gets too far away then the chain is broken, and all the agents in the queue behind it remain stuck with it.

### 2.6.2 Static Chain of Communication Linkage

The chain fountain effect-inspired method attempt showed the desirability to allow multiple agents to chain to a single agent. This would minimize or eliminate the issue of agents seeking to move through linked agents. The chain siphon method was used as a solution to this issue, although different implementations were initially tried. One such iteration used a static communication chain structure similar to the chain fountain method. Once an agent was linked it would stop attempting to link with other agents and would follow the agent it had linked with. This caused an issue around obstacles with sharp geometry. Agents moving out of a local minimum trap tend to hug the edge of the obstacle as they move around it. When multiple agents are linked to an agent rounding the edge of the obstacle the agents closer to the obstacle are pushed away from it due to obstacle repulsion while the agents further from the obstacle are pulled towards it to attempt to follow the agent they are linked with as closely as possible. This creates a "pinching" scenario where the outside agent is able to "edge out" the inside agent. While both agents are still trying to follow the agent they are linked with the inside agent is stalled from moving

until the outside agent moves far enough ahead to allow the collision avoidance to stop pushing the inside agent backwards. This can cause the inside agent to be pushed so far behind it is now trying to follow the agent it is linked with through the obstacle, running into the obstacle and remaining stuck. This issue displayed a need for dynamic generation of local chains of communication which would allow the agents to continuously assess the best agent for them to follow.

### 2.6.3 Queue Value Ceiling

Another issue arose in scenarios where the geometry caused an agent to lose the agent it was linked with. The agent would query it's neighbors but would only find the agent that is linked to it with a higher queue value. When this is the lowest queue value within it's neighbors then the agent thinks that agent is the furthest up in the queue and tries to follow it, placing itself in the queue behind it and having a queue value 1 higher than the neighbor. The neighbor now does the same with the agent and this repeats between them while they continue incrementing their queue values against each other. To avoid this issue a ceiling number was put on agents so that they went back to a default state when they reached it. This number cannot be too large or it will cause the agents to attempt to follow each other for extended lengths of time. The maximum number this should be is the number of agents in the swarm, as this would allow a single line of agents containing the entire swarm to move towards the goal if needed, although this number could be smaller if the size of the swarm is very large.

# Chapter 3

# Application In Simulated and Robotic Experiments

Our experiments consist of implementing our algorithm in a testbed with real robotic blimps and separately within a simulated environment using virtual agents for large numbers of samples. The robotic experiments run using the Lennard-Jones potential and chain siphon behaviors in a 2-D structure using all of the available agents in the lab. The simulations run using the Lennard-Jones potential, leader heuristic, and chain siphon behaviors in both 2-D and 3-D environments across a spread of the number of agents within the swarm.

## 3.1 Simulations

I choose to use the SCRIMMAGE [45] platform as the swarm simulator due to the modularity of the system allowing an easy transition between behaviors and its ease of scaling homogeneous swarms. For each type of behavior I set up both 2-D and 3-D simulations with an origin area the swarm initializes from, a goal area they attempt to find, and obstacles arranged in a cul-de-sac formation appropriate for the number of dimensions of each simulation. The obstacle is situated between the origin and the goal with the concave side facing the origin.

For the 2-D experiments, the 3 obstacles are arranged in a V formation as shown in Fig 3.2 and Fig 3.3. In the 3-D experiments, 5 obstacles are used as shown

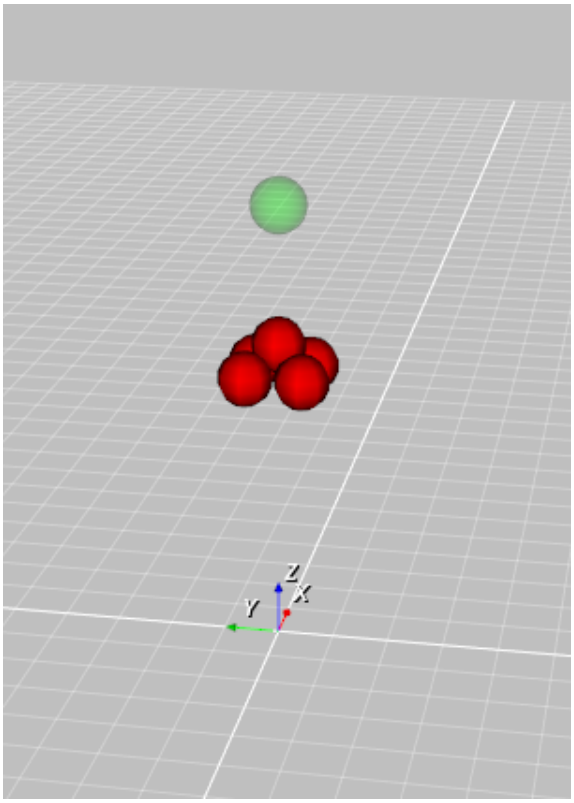



Figure 3.1: The environment of the 3-D SCRIMMAGE simulations.

in Fig 3.1. 4 of them are placed on the same plane in a square and the 5th one was placed above the empty middle space between them. Simulations were run for 100 simulated seconds. Most agents that reached the goal arrived within 30 seconds, the excess time allowed any unstable agents in the local minimum trap to destabilize and reach the goal or stabilize and remain stuck. If an agent had not reached the goal by the end of the simulation then it was assumed to be stable within the local minimum trap and counted as stuck.

For the each type of behavior in the 2-D and 3-D environments the simulations are divided into sets by the number of agents in the swarm, ranging from 1 to 50 agents. Each set of simulations for a specified number of swarm agents consist of 30 Monte Carlo simulations that track the number of agents that made it to the

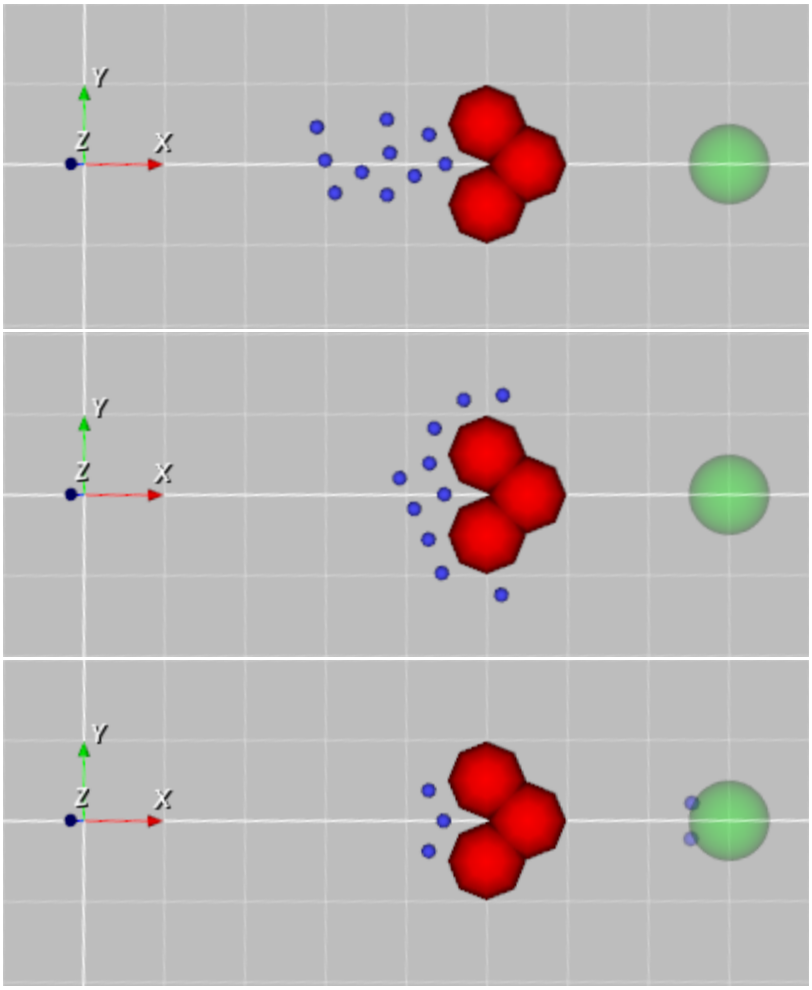



Figure 3.2: Sequence of frames from SCRIMMAGE simulating a swarm of agents using the Lennard-Jones Potential behavior. Agents flow around it like a viscoealstic fluid, although some remain stuck.

goal and the number that remain trapped. Their formation when they initialize is randomized for every simulation to examine the robustness of the chain siphon method as the way agents approach the local minimum trap contributes to how many can become stuck.

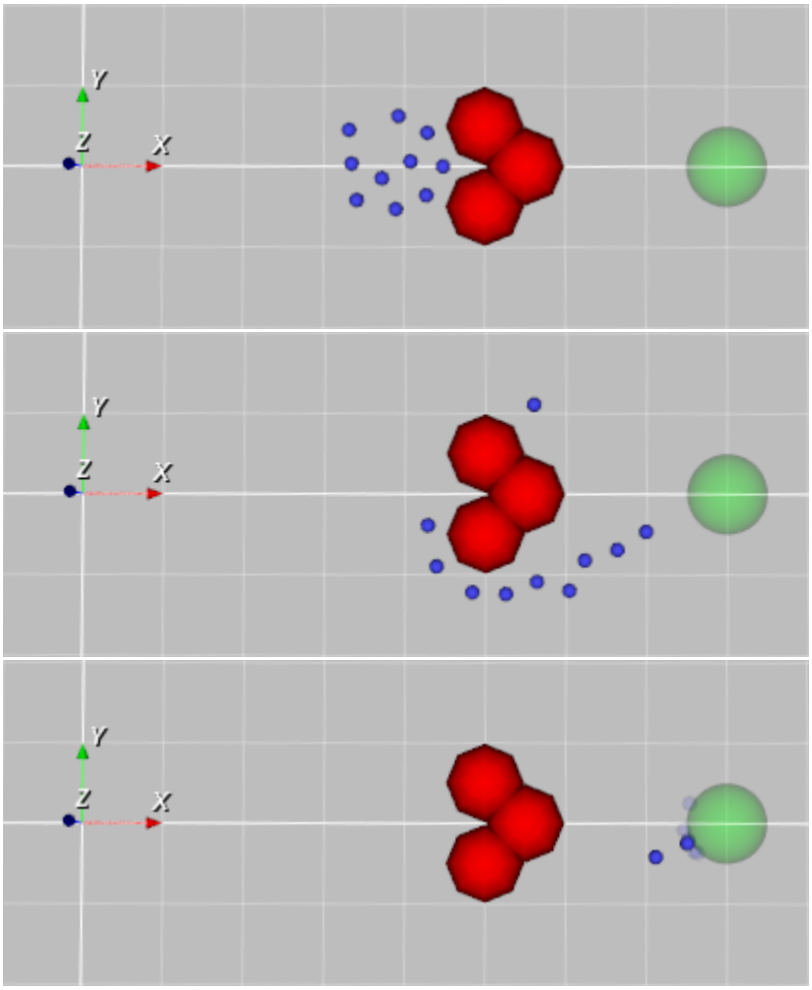



Figure 3.3: Sequence of frames from SCRIMMAGE simulating a swarm of agents using the Chain Siphon method behavior. Agents pull stuck agents out of traps.

## 3.2 Robotic Experiments

The GT-MABs localize using Vicon IR cameras and run using ROS. For compatibility with the previous waypoint controller the GT-MABs used, a "pseudo-velocity" controller was implemented. The controller takes the velocity vector calculated from equation 2.1 and adds it to the agent's current position, restricted to a specified maximum magnitude. This is updated every time the system receives new position data.

The experiments are set up similar to the simulations, with an initial starting area, a goal area, and a cul-de-sac obstacle between them. The workspace is artificially constrained in half in order to ensure the density of agents was concentrated enough to allow for a chain of communications to be established. The goal region is designated to be the half of the workspace past the obstacles to reduce the agents needed to create the chain of communication back to the local minima. Experiments are run in 2-D so that the Vicon system has all agents within its line of sight. The trials started once all agents were in the air within the starting area and ended once all agents not within the goal area were stabilized within the local minimum trap or when all agents reached the goal area. At the end of each trial the agents that had stabilized within the local minimum trap were counted as stuck. If agents malfunctioned during the run they were designated as "dead". If an agent died near the obstacle and trapped other agents the trial was deemed an outlier and the data was not considered.

Experiments started out with 12 agents which gradually reduced to 8 agents as they ran out of power or malfunctioned. Similar to the 2-D simulations, obstacles are represented by 3 repelling forces placed in a cul-de-sac like configuration. In the real world, stanchions are placed underneath the positions denoting the obstacles so that visibility was not obscured by large physical obstacles.

The experiment is run using ROS on a central computer. Vicon data for each agent is sent to the system, which determined the behavior for the agent and calculated the velocity. The controller makes continuously updating waypoints nearby the agent so it follows them. The inverse kinematics calculate the motor

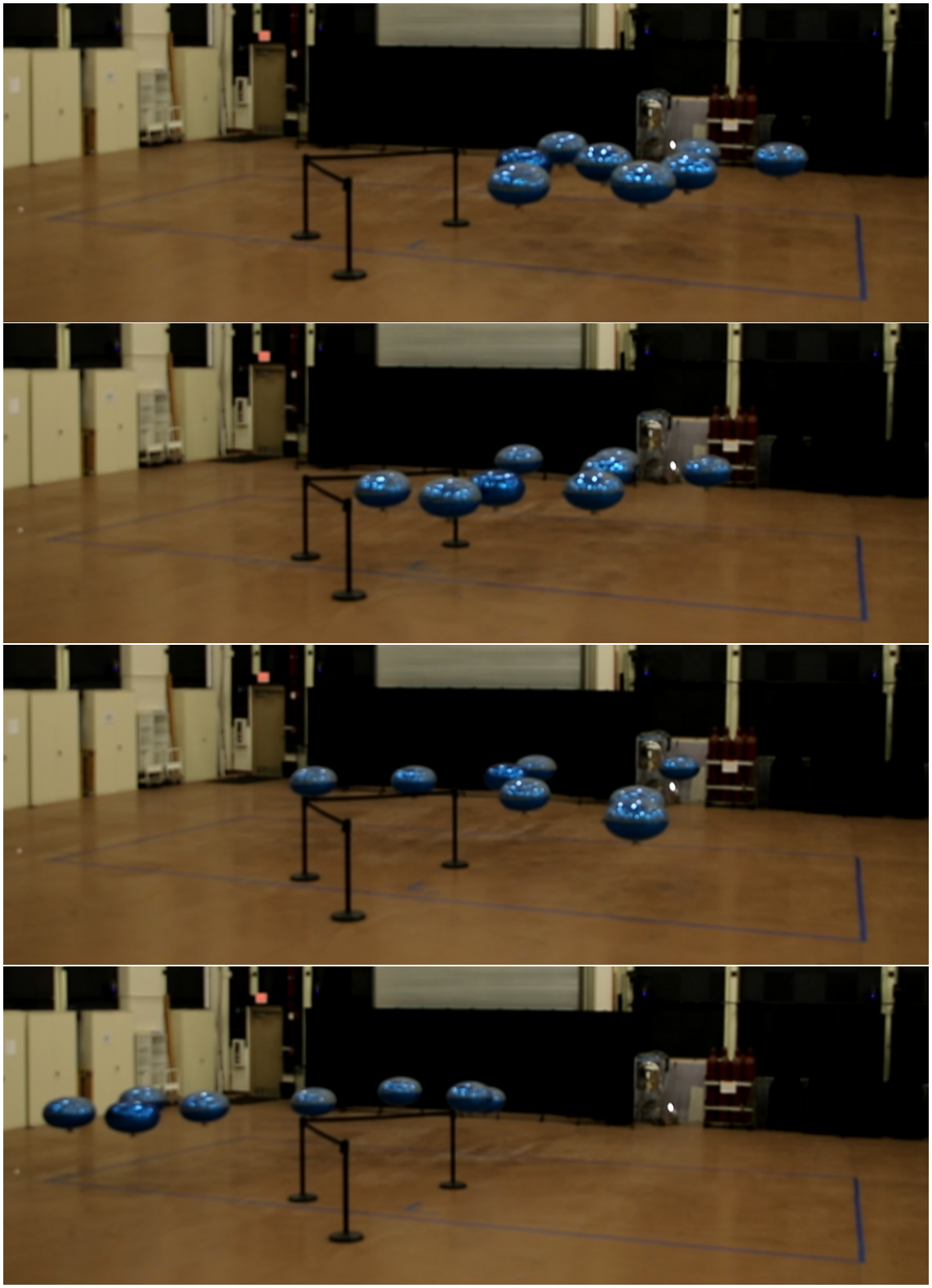


Figure 3.4: A hardware experiment implementing the Chain Siphon method. Agents are shown getting stuck in the local minimum then following their neighboring agents around the obstruction.

commands for each agent, which are sent over Xbee.

## 3.3 Real World Blimp Usage

The GT-MABs were chosen due to having a movement modality that is similar to the simulation agents, ease of implementation of software, and accessibility within the lab space. The real world implications of using the blimps presented drawbacks. While being a lighter-than-air vehicle allowed for long operating times between battery changes, this limited the payload weight each robot could carry. This issue was circumnavigated by removing the on-board sensing and using a central computer to direct the movements of the robots.

Using lighter-than-air vehicles also required them to be ballasted so they became nearly-neutrally buoyant. Blimps were weighted with clay until they could barely sink to the floor, ensuring they were near weightless but would not raise up to the ceiling without their motors. The near-neutral buoyancy subjected them to disturbances from the ambient air currents within the workspace. The environment used for the experiments was a spacious high bay room which generated a large amount of air circulation. Reducing circulation by eliminating air flow generation sources, such as turning off active HVAC systems within the room, contributed to improved control over blimp movement. There were also large high bay doors that lacked insulation producing a temperature differential. Large mobile walls were placed in front of them to reduce the effect they had on the air currents in the experiment space, and running experiments in the fall and spring when the temper-

ature differential between the outside and inside was reduced contributed to more accurate robot control.

In order to reduce the weight of onboard components, small motors were used which allowed for the use of batteries with smaller C ratings. This improved the amount of time the blimps were able to fly, but the small brushed motors had a lifespan of roughly 5 hours of operation. To combat motor degradation and malfunctioning, the motors were replaced at regular intervals to allow more consistent experimentation.

The envelopes used to hold helium for the blimps were made of Mylar. These held helium better than latex envelopes that were tested, but have reflective qualities. This reflectivity interfered with the Vicon system that was used, causing the IR cameras to detect reflections off of the balloons as opposed to just the reflective Vicon markers placed on the balloons. The balloons had their top half coated with a matte spray paint to reduce this source of noise for the Vicon system.

## 3.4 Parameter Selection

To choose proper parameters for each behaviour, the desired and undesired effects of each parameter were identified. For example, when $\epsilon$ is too large the attracting forces cause agents to stack up behind the local minimum trap, in a "top hat" formation. Having $\epsilon$ too small decreases the cohesion of the swarm. Designating the neighbor range too large causes agents to link to neighbors on the other side of obstacles, keeping them stuck in local minima traps. Making it

too small results in agents not being attracted to each other because the range at which they recognize a neighbor is placed within the collision avoidance range of the Lennard-Jones potential equation.

The parameters were manually selected and tested to find the high and low limits of the range where the undesired behavior begins to show. Then through iterative manual selection, parameter values were chosen that intuitively displayed the desired behavior.

The parameters within the simulation environment are unitless, but provide reference to the relative scale each parameter is given. The initial speed is set to 10 and the maximum speed was limited to 30. Weights for the control equation components for goal seeking, obstacle repulsion, and the Lennard-Jones potential behavior are set to 1, while the leader-heuristic coefficient is 10 and the siphon component weight is 75. The obstacle range is designated as 30 and obstacle repulsion is set to 1. The neighbor range is assigned 5, $\epsilon$ is 0.25, and collision range is set to 2.

In the physical experiments, the blimps roughly travel at 0.4 m/s at their fastest speed. The maximum speed set for the experiments is 0.75 of the max speed of the motors while the initial speed is 0.50 of the max speed. The range an agent senses obstacles was 1.5 m, and the obstacle repulsion value is set to 10. Sensing neighbors is limited to within 2.25 m, $\epsilon$ is given a value of 0.4, and the collision range is placed at 0.25 m. Weights of 1.0 are set for goal seeking and obstacle avoidance, while the Lennard-Jones potential weight is set to 0.05. During Lennard-Jones potential behavior experiments the chain siphon behavior weight is set to 0, but during chain siphon method experiments it is set to 20.0. A value of 50 is given to

a control behavior that keep the blimps within the designated workspace.

# Chapter 4

# Results

Based on the chosen parameters, our metrics are focused on how many agents remain stuck in minimum traps. For simulations we collect data correlating the number of agents within the swarm against the amount stuck. For hardware experiments we focus on collecting data from a dynamically challenging platform and how that affects the reliability of our results.

## 4.1 Simulations

For the Lennard-Jones potential behavior 2-D simulations in Figure 4.1, most of the agents become stuck every time when using 4 agents or less in the swarm. This number seems to increase logarithmically as the number of agents in the swarm increases. The ratio of trapped agents to non-stuck agents stays relatively low as the number of total agents increases. This is due to the local minimum trap becoming saturated with agents, and in order for more agents to get stuck they have to approach the stuck agents in a way that causes their cohesion and avoidance behavior to stabilize their position, while also not getting destabilized by any other agents passing near them. Having the swarm approach the area as a collective minimizes this stacking, as there are many agents in the vicinity of the stack that can destabilize it.

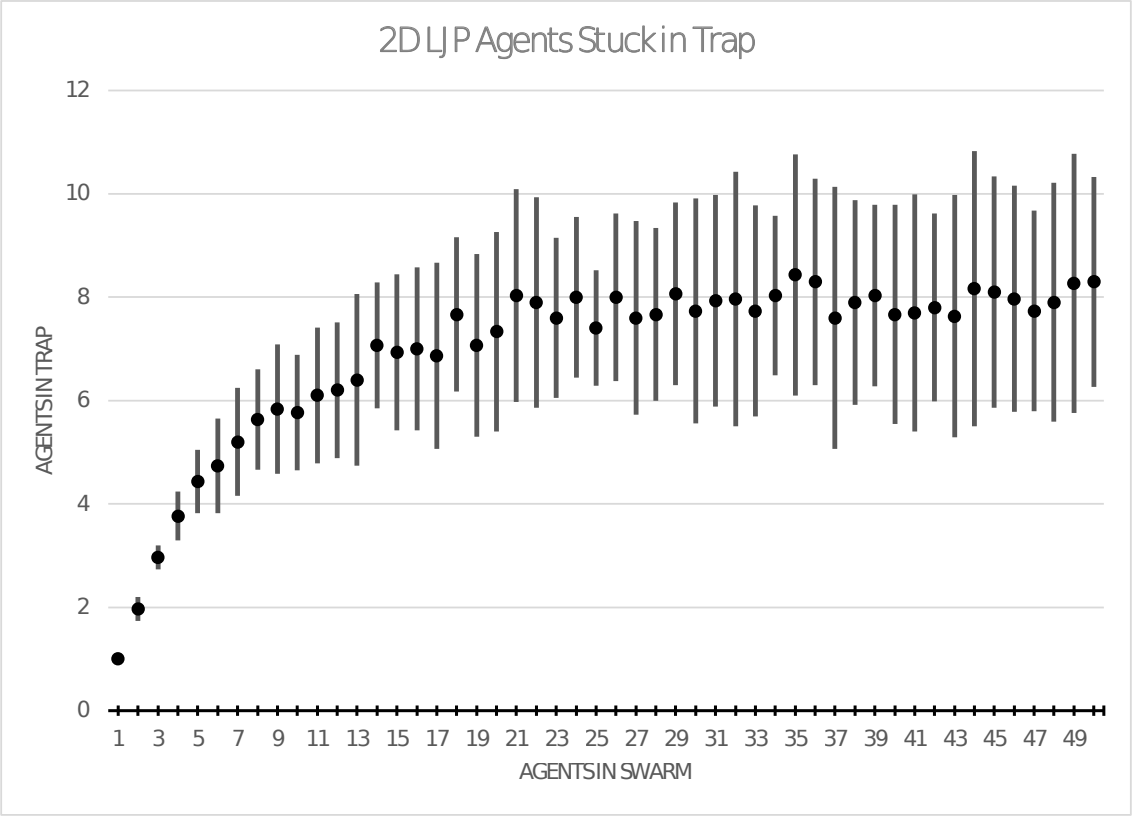



Figure 4.1: Data collected from the mass Monte Carlo LJP 2-D simulations.

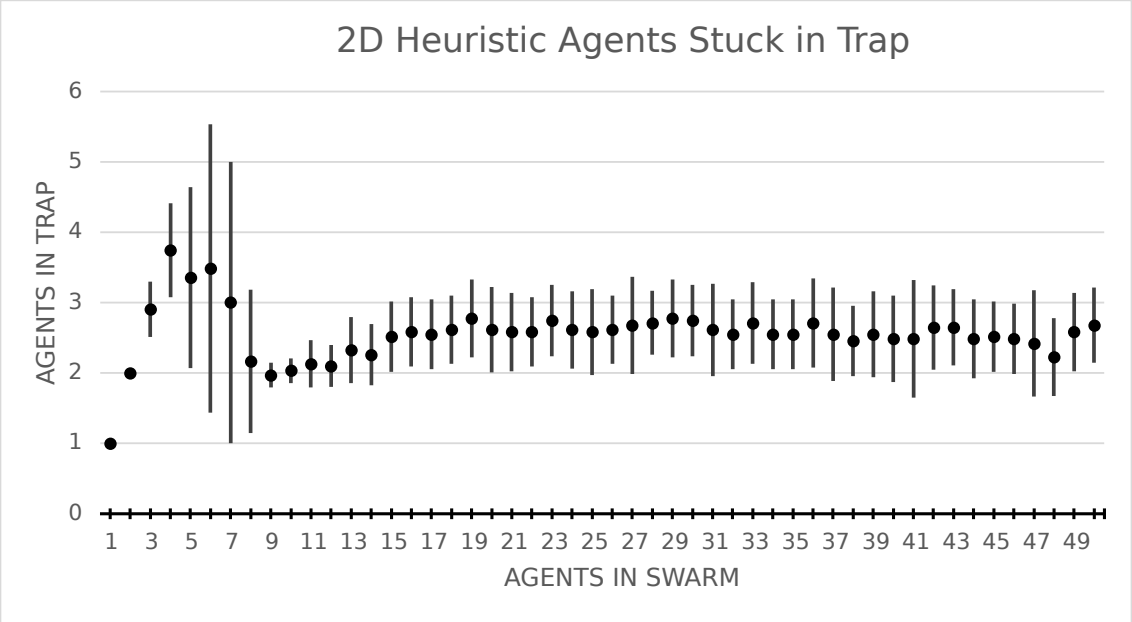



Figure 4.2: Data collected from the mass Monte Carlo leader heuristic 2-D simulations.

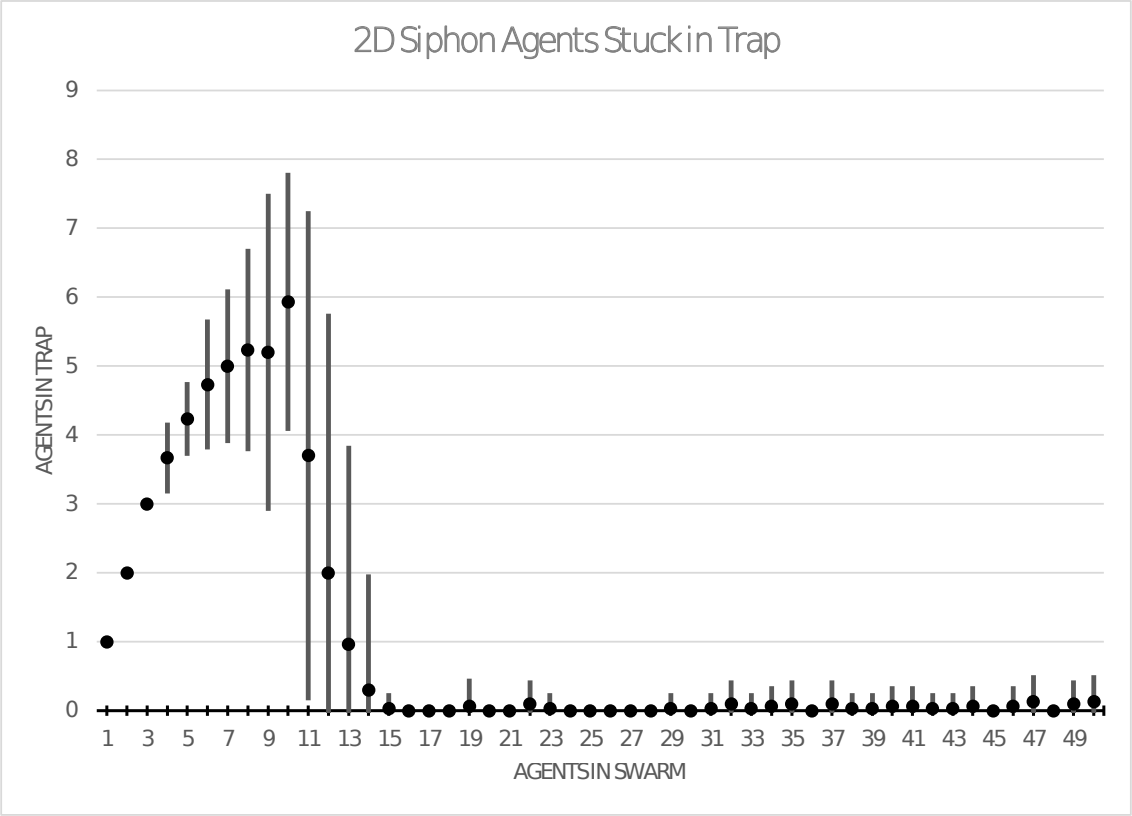



Figure 4.3: Data collected from the mass Monte Carlo chain siphon 2-D simulations.

The 3-D simulations in Figure 4.4 show similar results to the 2-D simulations. For simulations with 6 agents or less, almost all agents become trapped in every experiment. The number of stuck agents levels out at 9 agents as the number of swarm agents increases.

Using a leader heuristic behavior shows in Figure 4.2 the 2-D scenario that at once the swarm has grown to 5 agents there is a reduction in the number of agents getting stuck. This continues to decrease until an average of between 2 and 3 agents become stuck for every change in swarm size. The 3-D results in Figure 4.5 show the same behavior, but the change in behavior initiates around when there are 7 agents in the swarm, and decreases more slowly before it levels out.

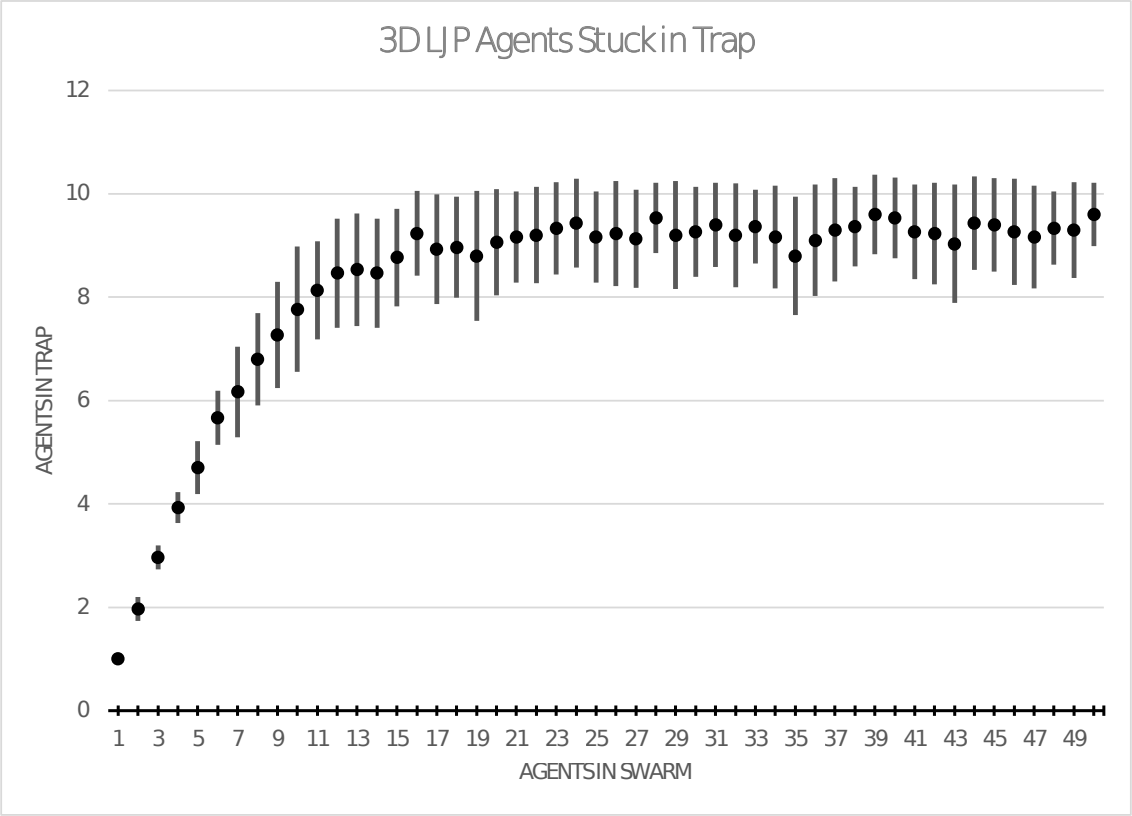



Figure 4.4: Data collected from the mass Monte Carlo LJP 3-D simulations.

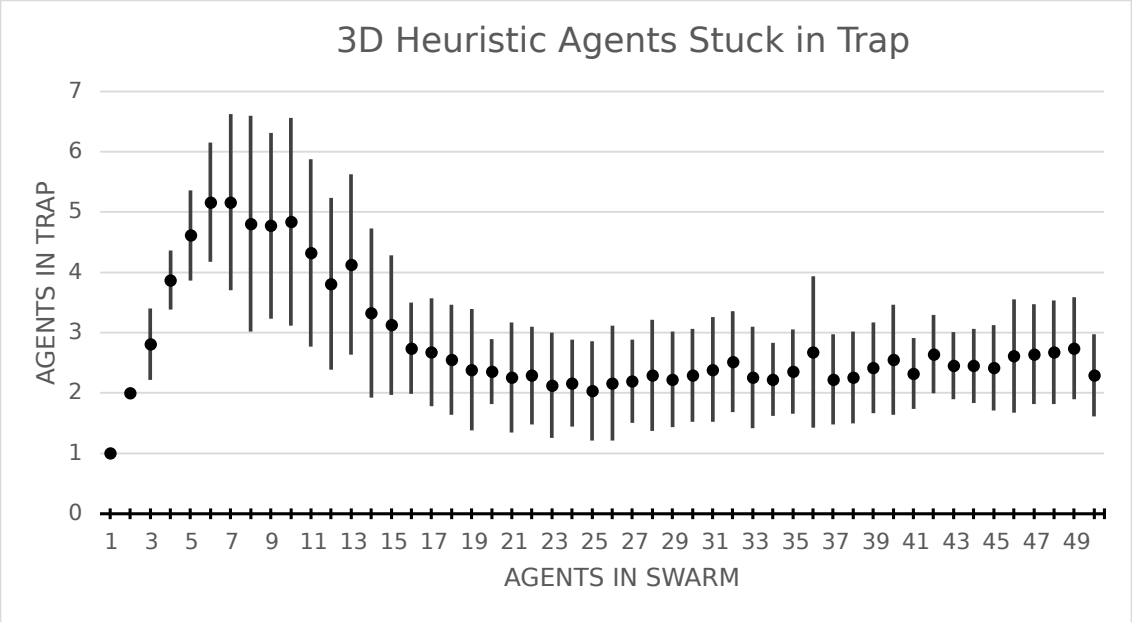



Figure 4.5: Data collected from the mass Monte Carlo leader heuristic 3-D simulations.

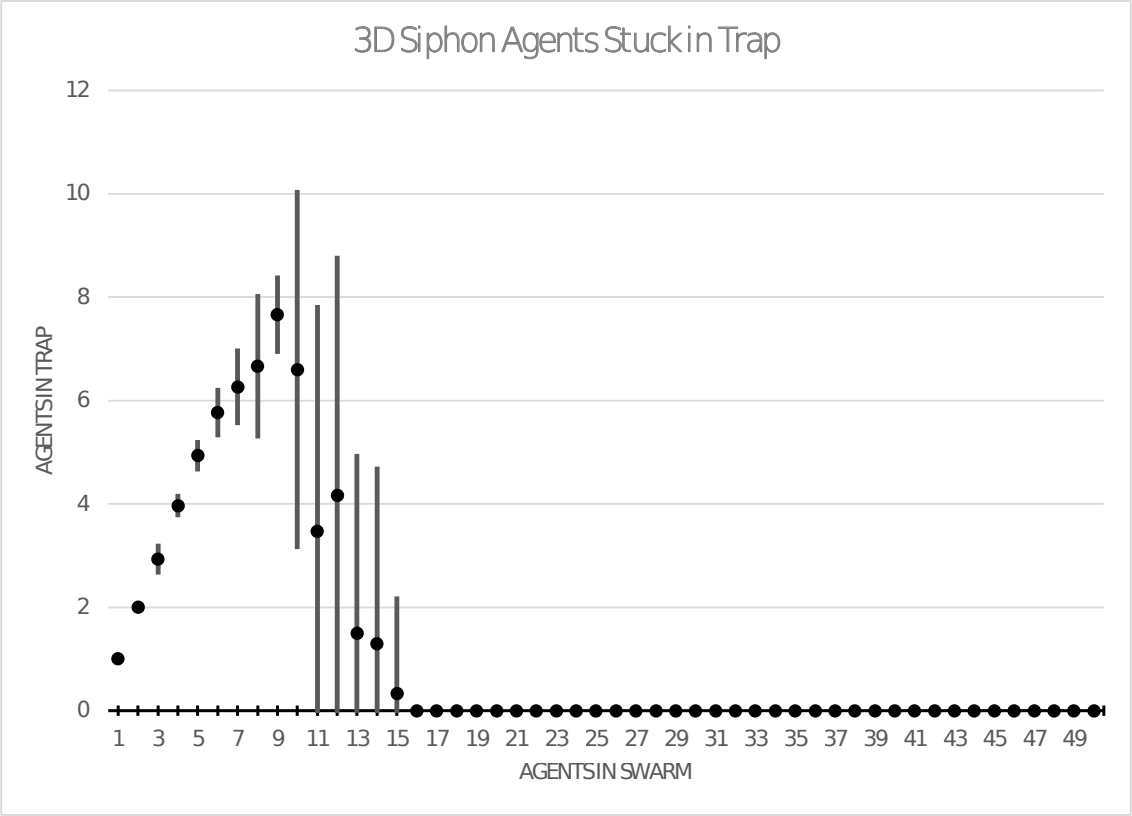



Figure 4.6: Data collected from the mass Monte Carlo chain siphon 3-D simulations.

The 2-D and 3-D chain siphon method simulations in Figures 4.3 and 4.6 show the same behaviors as the Lennard-Jones potential method simulations for a small number of agents in the swarm, as expected when the swarm does not have enough agents to establish the chain of communications back to agents that are trapped. In the 2-D case, once there are roughly 11 agents in the swarm it starts to exhibit the chain siphon effect, and the average number of agents stuck drastically decreases. At 15 agents it is able to consistently create a chain of communication and the average number of stuck agents stabilizes near zero.

For 3-D agents we see the chain siphon behavior begins to exhibit when the swarm is around 10 agents, as the average number of agents starts that get stuck

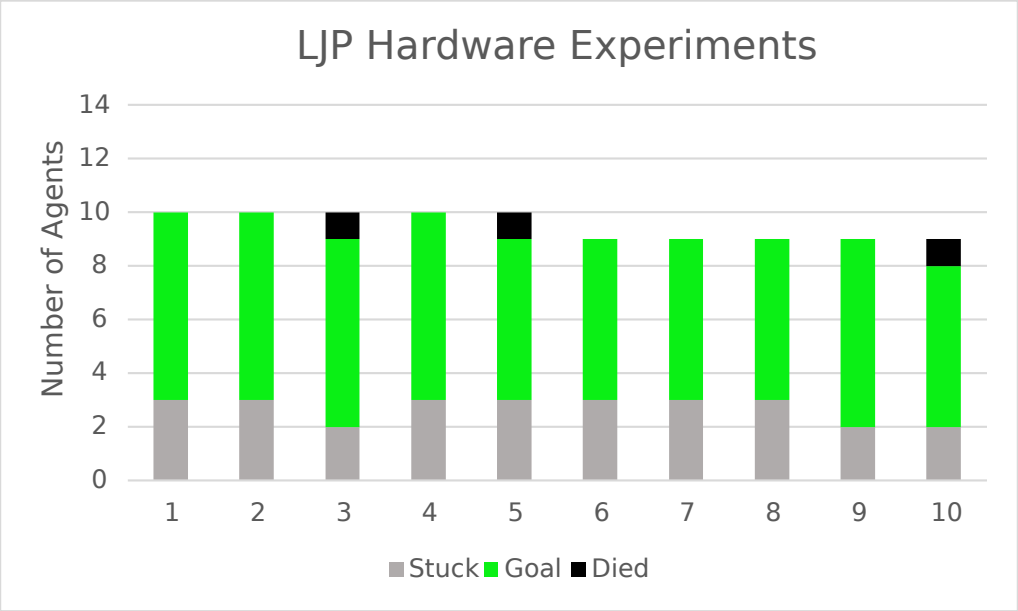
starts to decrease. This decrease continues until around 16 agents where it stabilizes at zero.

## 4.2 Robotic Experiments

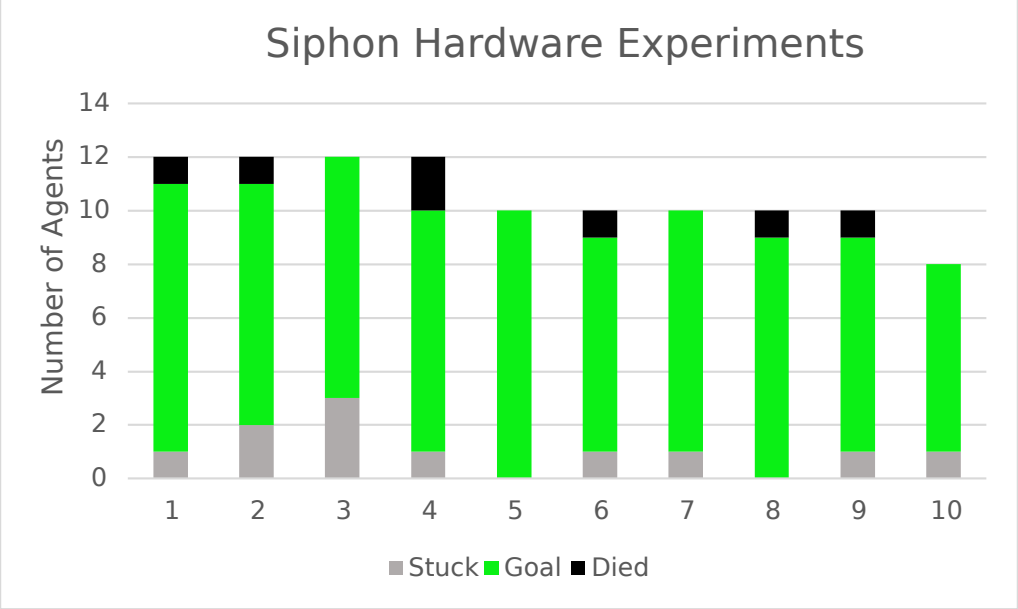
10 experiments are done for both the Lennard-Jones potential method and chain siphon method experiments. The average number of agents stuck during the Lennard-Jones potential experiments was 2.7 agents, with a standard deviation of 0.48. Generally 3 agents become stuck in a stable triangle configuration filling in the obstacle concavity. An average of 0.3 agents malfunction during those experiments with a standard deviation of 0.48.

The average number of agents that become stuck for the chain siphon experiments is 1.1 with a standard deviation of 0.88, and 0.7 agents malfunction during those samples with a standard deviation of 0.67. While the chain of communication initiates during all of the experiments, factors such as the blimps drifting due to atmospheric disturbances or their lack of fine motion control usually cause one to remain stuck in the local minima.

The number of total agents doesn't seem to have any noticeable effect when agents are reduced from 12 to 8 as they malfunction over the course of the experiments. This may not be the case if the number of agents reduces further though.




(a) Data collected from the Lennard-Jones Potential behavior hardware experiments




(b) Data collected from the Lennard-Jones Potential behavior hardware experiments

Figure 4.7: The mean (dots) and one standard deviation from the mean (bars) of agents stuck in the local minimum trap for each type of 3-D simulation.

# Chapter 5

# Conclusion

Using the results from the simulation and hardware experiments, evaluations in the performance of the chain siphon method can be made based on the differences between the Lennard-Jones potential, leader heuristic, and chain siphon methods. The similarities within the analyses between the simulations and hardware experiments can be used to evaluate how our results from the simulations translate to physical world experiments.

## 5.1 Discussion of Simulation Results

For all scenarios in which the agents are able to establish a link to the chain of local communications that the agent sensing the goal is a part of, almost all of those agents are pulled to the goal. There were several 2-D simulations in which agents would be pulled around either side of the obstacle and the final agent in the middle would be pulled to both sides causing the attractive forces to balance out and the agent would remain stuck. This could be an artifact of the size of the time steps in the simulation being too large, or the last agents in the chains on either side of the obstacle switching off having the smallest queue number due to more agents in their queue finding the goal area. This was not observed in the 3-D simulations, potentially due to the different geometry of the obstacle.

While this doesn't guaranty that all agents in the swarm reach the goal, since in some scenarios a chain of communication can not reach all the agents of the swarm, the chances of establishing the communications chain improves with an increasing number of swarm agents. As Figures 4.1 - 4.6 display, the drastic reduction in the mean number of agents that become stuck in the chain siphon method simulations after the number of agents has increased to 10 agents displays the capabilities of the method. This can be contrasted to the Lennard-Jones potential method where at that point the number of stuck agents is still increasing and the leader heuristic method where the number of stuck agents has leveled off between 2 and 3.

The reduction in performance in the leader heuristic method compared to the work from Mabrouk et al. [23] can be attributed to the domain that it is being used in. Not having access to global communication limits the amount of stuck agents that are able to be pulled from local minima, and the implementation in this work still allows global knowledge of when an agent is outside of a local minima area. The chain siphon method does not require these types of global capabilities, allowing it to be better suited in communication and sensor range limited environments and within distributed systems.

An analysis of the standard deviation through the simulations reveals insights about the reliability of the chain siphon method. The standard deviation expands more in the 2-D chain siphon method simulations at 9 agents compared to the Lennard-Jones potential method data, showing the chain siphon method starting to influence the agents in some trials during that set of simulations. The standard deviation remains large until there are 15 agents in the swarm, showing between

9 and 15 agents the method is unreliable for this simulation setup. At 15 agents the number of stuck agents has approached near 0 and remains there as the swarm number scales up with minor relative variations. From this it can be seen that there is a critical threshold for the number of agents needed in the swarm before reliability can be assumed.

An analysis of the 3-D chain siphon method simulations shows the same conclusions at similar numbers of agents. Once the critical threshold for the chain siphon method's reliability has been reached it shows even more reliable results. After the system grows to 16 agents the mean and standard deviation of stuck agents are 0 for this simulation setup, even as the number of agents continues to scale.

## 5.2 Discussion of Hardware Results

The differences with the real world experiments arise from the robots chosen for the experiments. Due to the blimps' buoyancy being nearly neutral, they are heavily influenced by the atmospheric airflow in the environment. This causes variations in their positions in contrast to the simulation which has no unexpected variations in positions or velocities. These variations create situations where the blimps would start the queue chain and one of them may drift outside the range of the agent it had previously been linked with. This would be less of a problem for robots with finer controls or that could move along the ground, although a similar situation could arise from the use of differential drive vehicles or other vehicles with nonholonomic dynamics.

Despite the differences between the physical experiments and simulations, similarities can be drawn by examining the reduction in the number of agents that are stuck when using the chain siphon method compared to the Lennard-Jones potential method. The median number of stuck agents during Lennard-Jones potential behavior experiments was 3, and the median for chain siphon method was 1. While most chain siphon method experiments didn't cause all agents to escape, the reduction in the number of stuck agents was a desirable result and showed positive outcomes even when using imprecise moving platforms such as the GT-MABs.

## 5.3 Future Work

Future work could be done on this algorithm expanding the robustness of it's implementation. Developing a structure to use this without a static goal area would improve functionality. For example, if an agent moving through a space could sense other agents stuck, then a dynamic temporary goal could be placed outside of the local minima trap to allow the agent to pull out the stuck agents. Another addition could be implementing biasing factors so that the swarm does not need a complete communications tether between the goal and the agents in the local minima trap in order for them to escape from the trap. Biasing heuristics such as following agents further from obstacles or ones with a higher velocity gradient, implying there are agents stuck and agents moving, could aid in achieving this. Also modifying the behavior so it can be used on vehicles with nonholonomic controls or heterogeneous swarms would improve the capabilities of the algorithm. It could be beneficial to

run real world experiments in 3-D and on a truly distributed-capable hardware systems, but the current platform restrains us from this and would require further development to access this type of functionality.

# Bibliography


[1] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, 1986.

[2] S. Chung, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar. A survey on aerial swarm robotics. *IEEE Transactions on Robotics*, 34(4):837–855, 2018.

[3] A. Becker, G. Habibi, J. Werfel, M. Rubenstein, and J. McLurkin. Massive uniform manipulation: Controlling large populations of simple robots with a common input signal. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 520–527, 2013.

[4] Aaron Becker, Erik D. Demaine, Sándor P. Fekete, Golnaz Habibi, and James McLurkin. Reconfiguring massive particle swarms with limited, global control. In Paola Flocchini, Jie Gao, Evangelos Kranakis, and Friedhelm Meyer auf der Heide, editors, *Algorithms for Sensor Systems*, pages 51–66, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[5] Aaron Becker, Cem Onyuksel, Timothy Bretl, and James McLurkin. Controlling many differential-drive robots with uniform control inputs. *The International Journal of Robotics Research*, 33(13):1626–1644, 2014.

[6] Hamsa Balakrishnan. Control and optimization algorithms for air transportation systems. *Annual Reviews in Control*, 41:39 – 46, 2016.

[7] P. K. Menon, G. D. Sweriduk, and K. D. Bilimoria. New approach for modeling, analysis, and control of air traffic flow. *Journal of Guidance, Control, and Dynamics*, 27(5):737–744, 2004.

[8] S. James, R. Raheb, and A. Hudak. Uav swarm path planning. In *2020 Integrated Communications Navigation and Surveillance Conference (ICNS)*, pages 2G3–1–2G3–12, 2020.

[9] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, 1(1):269–271, December 1959.

[10] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, July 1968.

[11] Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, 1998.

[12] L. E. Kavraki, P. Svestka, J. . Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[13] L. Barnes, W. Alvis, M. Fields, K. Valavanis, and W. Moreno. Swarm formation control with potential fields formed by bivariate normal functions. In *2006 14th Mediterranean Conference on Control and Automation*, pages 1–7, June 2006.

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

[15] Qifeng Chen, Sandor Veres, Yaonan Wang, and Yunhe Meng. Virtual spring-damper mesh-based formation control for spacecraft swarms in potential fields. *Journal of Guidance, Control, and Dynamics*, 38:1–8, 02 2015.

[16] C. Bentes and O. Saotome. Dynamic swarm formation with potential fields and a* path planning in 3d environment. In *2012 Brazilian Robotics Symposium and Latin American Robotics Symposium*, pages 74–78, Oct 2012.

[17] H. Chiang, N. Malone, K. Lesser, M. Oishi, and L. Tapia. Path-guided artificial potential fields with stochastic reachable sets for motion planning in highly dynamic environments. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2347–2354, May 2015.

[18] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, Nov 1995.

[19] Rainer Palm and Abdelbaki Bouguerra. Particle swarm optimization of potential fields for obstacle avoidance. 09 2013.

[20] D. Zhou, Z. Wang, and M. Schwager. Agile coordination and assistive collision avoidance for quadrotor swarms using virtual structures. *IEEE Transactions on Robotics*, 34(4):916–923, Aug 2018.

[21] John H. Reif and Hongyan Wang. Social potential fields: A distributed behavioral control for autonomous robots. *Robotics and Autonomous Systems*, 27(3):171 – 194, 1999.

[22] Cezary Kownacki and Leszek Ambroziak. Local and asymmetrical potential field approach to leader tracking problem in rigid formations of fixed-wing uavs. *Aerospace Science and Technology*, 68:465–474, 09 2017.

[23] Mohamed Mabrouk and C.R. McInnes. Swarm robot social potential fields with internal agent dynamics. *International Conference on Aerospace Sciences and Aviation Technology*, 12, 05 2007.

[24] H. E. Espitia and J. I. Sofrony. Path planning of mobile robots using potential fields and swarms of brownian particles. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, pages 123–129, June 2011.

[25] Matteo Guerra, Denis Efimov, Gang Zheng, and Wilfrid Perruquetti. Avoiding local minima in the potential field method using input-to-state stability. *Control Engineering Practice*, Volume 55:174–184, 07 2016.

[26] R. Gayle, W. Moss, M. C. Lin, and D. Manocha. Multi-robot coordination using generalized social potential fields. In *2009 IEEE International Conference on Robotics and Automation*, pages 106–113, 2009.

[27] Fethi Matoui, B. Boussaid, and Abdelkrim Mohamed Naceur. Local minimum solution for the potential field method in multiple robot motion planning task. pages 452–457, 12 2015.

[28] Suranga Hettiarachchi, William Spears, Derek Green, and Wesley Kerr. Distributed agent evolution with dynamic adaptation to local unexpected scenarios. volume 3825, pages 245–256, 09 2005.

[29] J. E. Jones. On the Determination of Molecular Fields. II. From the Equation of State of a Gas. *Proceedings of the Royal Society of London Series A*, 106(738):463–477, Oct 1924.

[30] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph.*, 21(4):25–34, August 1987.

[31] Suranga Hettiarachchi and William M. Spears. Distributed adaptive swarm for obstacle avoidance. *International Journal of Intelligent Computing and Cybernetics*, 2(4):644–671, 2009.

[32] Nitin Sydney, Derek A. Paley, and Donald Sofge. Physics-inspired motion planning for information-theoretic target detection using multiple aerial robots. *Autonomous Robots*, 41(1):231–241, Jan 2017.

[33] R. Rajan, M. Otte, and D. Sofge. Novel physicomimetic bio-inspired algorithm for search and rescue applications. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, Nov 2017.

[34] Wesley Kerr. *Gas-Mimetic Swarms for Surveillance and Obstacle Avoidance*, pages 193–221. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[35] H. G. Tanner, A. Jadbabaie, and G. J. Pappas. Stable flocking of mobile agents, part i: fixed topology. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*, volume 2, pages 2010–2015 Vol.2, Dec 2003.

[36] H. G. Tanner, A. Jadbabaie, and G. J. Pappas. Stable flocking of mobile agents part ii: dynamic topology. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*, volume 2, pages 2016–2021 Vol.2, Dec 2003.

[37] H. G. Tanner, A. Jadbabaie, and G. J. Pappas. Coordination of multiple autonomous vehicles. In *2003 IEEE Mediterranian Conf. Control and Automation (MED)*, 2003.

[38] Sungjin Cho, Vivek Mishra, Qiuyang Tao, Paul Varnell, Matt King-Smith, Aneri Muni, Weston Smallwood, and Fumin Zhang. Autopilot design for a class of miniature autonomous blimps. pages 841–846, 08 2017.

[39] S. van der Zwaan, A. Bernardino, and J. Santos-Victor. Vision based station keeping and docking for an aerial blimp. In *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113)*, volume 1, pages 614–619 vol.1, Oct 2000.

[40] S. B. V. Gomes and J. G. Ramos. Airship dynamic modeling for autonomous operation. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, volume 4, pages 3462–3467 vol.4, May 1998.

[41] Qiuyang Tao, Jaeseok Cha, Mengxue Hou, and Fumin Zhang. Parameter identification of blimp dynamics through swinging motion. pages 1186–1191, 11 2018.

[42] Tristan Schuler, Daniel Lofaro, Loy Mcguire, Alexandra Schroer, Tony Lin, and Donald Sofge. A study of robotic swarms and emergent behaviors using 25+ real-world lighter-than-air autonomous agents (lta 3 ). 10 2019.

[43] Jason Gibson, Tristan Schuler, Loy McGuire, Daniel M. Lofaro, and Donald Sofge. Swarm and multi-agent time-based a* path planning for lighter-than-air systems. *Unmanned Systems*, 08(03):253–260, 2020.

[44] Divya Srivastava, Daniel Lofaro, Tristan Schuler, and Donald Sofge. Gesture-based interface for multi-agent and swarm formation control. 11 2019.

[45] Kevin DeMarco, Eric Squires, Michael Day, and Charles Pippin. Simulating collaborative robots in a massive multi-agent game environment (SCRIMMAGE). In *Int. Symp. on Distributed Autonomous Robotic Systems*, 2018.