Viscoelastic Fluid-Inspired Swarm Behavior to Reduce Susceptibility to Local Minima: The Chain Siphon Algorithm

Loy McGuire, Tristan Schuler, Michael Otte, and Donald Sofge, *Senior Member, IEEE*

***Abstract*—We present a novel distributed robotic swarm algorithm inspired by the open channel siphon phenomenon displayed in certain viscoelastic fluids. Self-siphoning viscoelastic fluids are often able to mitigate the trapping effects of local minima in the environment. Using a similar strategy, our algorithm enables a robot swarm to mitigate the trapping effects of local minima in potential fields. 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 traps and often eliminating them. We perform hardware experiments using the Georgia Tech Miniature Autonomous Blimp (GT-MAB) aerial robotic platforms as well as Monte Carlo simulations conducted in the Simulating Collaborative Robots in Massive Multi-Agent Game Execution (SCRIMMAGE) simulator. We compare the new method to other potential field based swarm behaviors that both do and do not incorporate local minima fixes. The distributed algorithm generates self-siphoning behavior within the robotic swarm, and this reduces its susceptibility to local minima.**

***Index Terms*—Swarm robotics, distributed robot systems, planning under uncertainty.**

I. Introduction

SWARMS are multi-agent systems that provide benefits such as collective action, multitasking, and redundancy. They are useful for tasks such as collective object transportation and exploration. Swarms may contain an arbitrarily large number of agents. A defining characteristic of swarm algorithms is continued functionality as swarm size is increased by orders of magnitude.

Manuscript received August 25, 2021; accepted October 25, 2021. Date of publication November 17, 2021; date of current version December 28, 2021. This letter was recommended for publication by Associate Editor Kirstin Hagelskjaer and Editor M. Ani Hsieh upon evaluation of the reviewers' comments. This work was supported by the Office of Naval Research (ONR) under Agreement #N0001421WX00142. *(Corresponding author: Loy McGuire.)*

Loy McGuire is with the Distributed Autonomous Systems Group, U.S. Naval Research Laboratory, Washington, DC 20375 USA (e-mail: loy.mcguire@nrl.navy.mil).

Tristan Schuler is with the Distributed Autonomous Systems Group, U.S. Naval Research Laboratory, Arlington, Virginia 22204 USA (e-mail: tristan.schuler@nrl.navy.mil).

Michael Otte is with the University of Maryland, College Park, Maryland 20742 USA (e-mail: ottemw@gmail.com).

Donald Sofge is with the Distributed Autonomous Systems Group, U.S. Naval Research Laboratory, Washington, DC 20375 USA (e-mail: don.sofge@nrl.navy.mil).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2021.3128705, provided by the authors.

Digital Object Identifier 10.1109/LRA.2021.3128705

One ramification of this scalability requirement is that artificial swarms are often organised as locally communicating distributed systems, and swarm algorithms are designed to use local decision making and local message passing. This organisation facilitates scalability by reducing both computational complexity and communication requirements, but does not guarantee algorithmic completeness. In contrast, centralized systems that provide algorithmic completeness require global coordination and communication, which can cause computational intractability and communication bandwidth saturation for large numbers of agents.

A variety of methods have been proposed to direct the movement of swarms. Potential field methods associate all positions within a workspace with a magnitude. The gradient of the potential field is then interpreted as a vector field, imposed on the workspace, and used to control an agent. Potential fields were originally used by Khatib to control robot manipulator arms [2], but have also been widely used to control swarms of agents [3]–[7].

It is well known that potential fields may contain local minima that can trap agents (illustrated in Fig. 1). A local minimum occurs when a low potential area is surrounded by higher potentials, causing a local sink (point of attraction) in the vector field. In nature, certain viscoelastic fluids overcome local minima using a self-siphoning phenomena: once a viscoelastic fluid starts pouring out of a container, internal viscoelastic forces continue to pull more and more of the material over the container wall (see Fig. 1). We present the *Chain Siphon Algorithm*, a distributed swarm algorithm inspired by self-siphoning viscoelastic fluids.

The Chain Siphon Algorithm relies on an assumption that a large number of agents exist, and then leverages distributed sensing and local communication to achieve a self-siphoning effect. First, as more agents become stuck in a local minimum trap, the trap eventually fills with agents such that additional agents 'spill over' and find the goal. Second, assuming many agents exist, a queue of local communication is formed from the agents in the goal to the agents in the local minima traps. Third, robot order in the queue is used to alter each agent's potential field such that the swarm is drawn out of the local minima and toward the goal.

Running the Chain Siphon Algorithm on all agents simultaneously causes an emergent behavior that is similar to a viscoelastic fluid exhibiting the open channel siphon effect. This siphoning behavior allows agents that successfully move around obstacles to pull neighboring agents out of local minima traps. We test

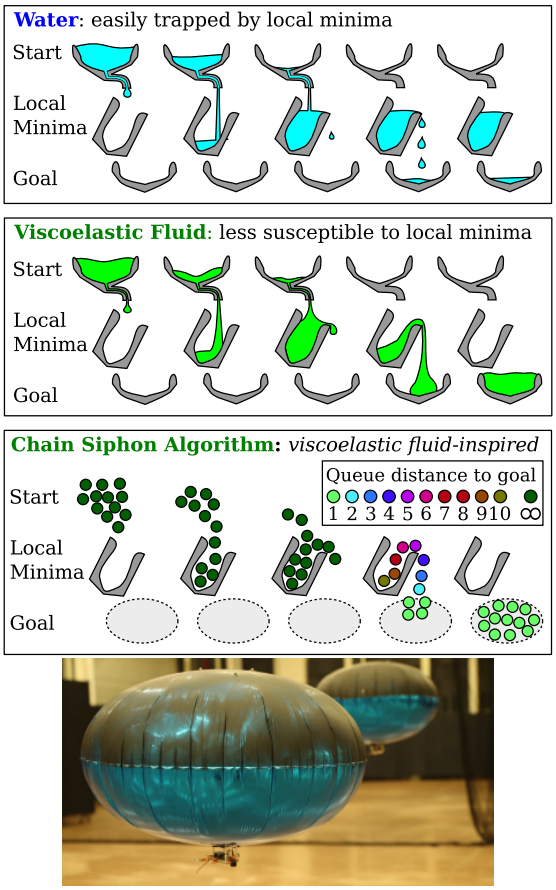

Fig. 1. Water is trapped by a local minimum as it falls downward (top panel). A viscoelastic fluid is less susceptible to local minima due to its self siphoning behavior (second panel). Our distributed chain siphon algorithm, inspired by viscoelastic fluids, uses a similar siphoning behavior to enable robots swarms to overcome the local minima that affect potential field methods (third panel). It is advantageous when at least one robot reaches the goal. Bottom [1]: We test our algorithm on a swarm of robot blimps.

the Chain Siphon Algorithm in simulation (Fig. 6) and with real-world robotic experiments using a swarm of GT-MABs (Fig. 4).

The rest of the paper is organized as follows: Section II discusses related work. Section III details the Chain Siphon Algorithm, other algorithms we compare to, and necessary subroutines. Our experiments are presented in Section IV, and the results from those experiments are discussed in Section V. Conclusions appear in Section VI.

II. Related Work

Potential fields have long been used to control multi-agent and swarm robotic systems, dating back to at least 1989 [8]. For example, mathematical relationships such as bivariate normal functions [3], sigmoid, and normal functions [4] are used to keep robots in a desired formation as the swarm follows a global trajectory. A mass-spring-damper model is used to control agent-agent interactions in spacecraft swarms [5]. Other work has combined A* global planning with potential fields for formation control and obstacle avoidance [6], or used sampling-based motion planning to guide agents to intermediate goals while associating repulsive forces with dynamic obstacles [7]. In contrast, our method does not require the swarm as a whole to follow a trajectory. Each agent individually acts and interacts with its surroundings, which causes a beneficial global emergent behavior to form across the swarm.

Potential fields are also useful for balancing multiple objectives using a single control equation. For example, Particle Swarm Optimization [9], has been used to optimize which behaviors are most prevalent in control schemes incorporating multiple behaviors [10]. Competing objectives of formation control and collision avoidance are considered in [11]. Virtual agents, defined *a priori*, are used as 'leaders' for swarm control in [12]. The Chain Siphon Algorithm shares similarities with a leader-follower method; however, the Chain Siphon Algorithm's queue order is based on online swarm position, and not defined *a priori*.

Social potential fields [13] are a class of potential field methods in which agents dynamically influence their neighbors. The Chain Siphon Algorithm can be considered a social potential field method.

A variety of multi-agent methods have been used to avoid local minima traps. Dynamically generating the vector fields that control agent movement can prevent agents from being trapped in local minima in certain scenarios [3], [4]. Agents able to recognize local minima can increase the potentials of their dynamic internal fields [14] to prevent becoming trapped. Mabrouk *et al.* use a swarm leader heuristic where *a priori* global knowledge of local minima helps agents escape traps through random motion [14]. Vortex and Brownian motion inspired behavior enable agents to escape local minima by increasing their Brownian motion [15]. Matouri *et al.* assign agents' priorities and then augment speed to avoid collisions [16]. In contrast to the aforementioned work, the Chain Siphon Algorithm actively overcomes traps without prior knowledge of the environment, and without stochastic movement.

Our work is closely related to physicomimetic (physics mimicking) [17] ideas such as DAEDALUS [18], as well as biologically inspired ideas such as Boids flocking algorithm [19]. DAEDALUS uses the Lennard-Jones potential (LJP) equation to calculate collision avoidance and swarm cohesion forces [20]. Boids algorithm causes agents to steer away from neighbors, towards the average heading of neighbors, and towards the center of mass of local agents. Tanner *et al.* creates a Boids-like flocking behavior using separation, alignment, and cohesion elements but not the Lennard-Jones potential function [21]–[23].

We used the GT-MABs as the robots for our experiments. The dynamic model for the blimps was first presented in [24], [25], and a control system was built around their models for actuation.

III. Methodology

Details of the Chain Siphon Algorithm are discussed in (Section III-A-III-B). The control equations and vectors used for

Algorithm 1: Chain Siphon Algorithm (Runs in parallel on all agents)

1: $i \leftarrow$ ID of agent currently running algorithm
2: $n_s \leftarrow$ Total number of agents in the swarm
3: $d_{min} \leftarrow$ Maximum neighbor range
4: **loop**
5: $\quad \mathbf{C_{siphon}} = \mathbf{0}$
6: $\quad q_i = n_s$ % Default position in queue
7: $\quad$ **if** $senseGoal()$ **then**
8: $\quad\quad q_i = 1$
9: $\quad q_{min} = n_s$
10: $\quad N_i = queryNeighbors()$
11: $\quad$ **for all** $j \in N_i$ **do**
12: $\quad\quad \mathbf{v_{ij}} = \mathbf{x_j} - \mathbf{x_i}$ % Rel. position vector
13: $\quad\quad d_{ij} = \|\mathbf{v_{ij}}\|$
14: $\quad\quad$ **if** $q_j < q_{min}$ **then**
15: $\quad\quad\quad q_{min} = q_j$
16: $\quad\quad\quad d_{min} = d_{ij}$
17: $\quad\quad\quad \mathbf{v_{link}} = \mathbf{v_{ij}}$
18: $\quad\quad$ **if** $q_j = q_{min}$ **and** $d_{ij} < d_{min}$ **then**
19: $\quad\quad\quad d_{min} = d_{ij}$
20: $\quad\quad\quad \mathbf{v_{link}} = \mathbf{v_{ij}}$
21: $\quad$ **if** $q_{min} < q_i$ **then**
22: $\quad\quad q_i = q_{min} + 1$
23: $\quad\quad \mathbf{C_{siphon}} = \frac{\mathbf{v_{link}}}{\|\mathbf{v_{link}}\|}$
24: $\quad$ **if** $q_i > n_s$ **then**
25: $\quad\quad q_i = n_s$
26: $\quad broadcast(i, q_i)$
27: $\quad$ **return** $\mathbf{C_{siphon}}$

goal seeking, obstacle avoidance, and the Lennard-Jones potential flocking behavior are discussed in (Section III-C-III-D). A description of a leader heuristic method to contrast with the Chain Siphon Algorithm is in (Section III-D).

A. Notation

The total number of agents is denoted n_s. The queue value of a specific agent i is q_i, and the minimum queue value found between agent i and any agent j within the set N_i of agents neighboring agent i is q_{min}. The distance between agent i and object j is d_{ij}, where an *object* may be another agent or an obstacle, depending on context. The minimum distance between an agent and the neighboring agents for which $q_j = q_{min}$ is denoted as d_{min}. The global position vector of agent i is $\mathbf{x_i}$ and can be used to compute the relative position vector $\mathbf{v_{ij}}$ between agent i and object j. The vector that links the agent with the neighbor it is following in the queue is denoted by $\mathbf{v_{link}}$. The control equation calculates a vector $\mathbf{V}$ using weights w_i and control vectors $\mathbf{C_i}$. The goal position is denoted $\mathbf{x_{goal}}$. The set of obstacles sensed by an agent is O_i.

Parameter a defines the magnitude of the repulsion force from obstacles. LJP_r is the intermolecular potential, σ is the distance at which $\|\mathbf{V}\|$ is zero, r is the distance from atom i to atom j, and ϵ is a parameter specifying the strength of their interactions. Parameters b and c define the attractive and repulsive movement of the agent. The leader heuristic parameter e determines the magnitude of the bias towards the leader for neighboring agents. The values of these parameters are found in Appendix A.

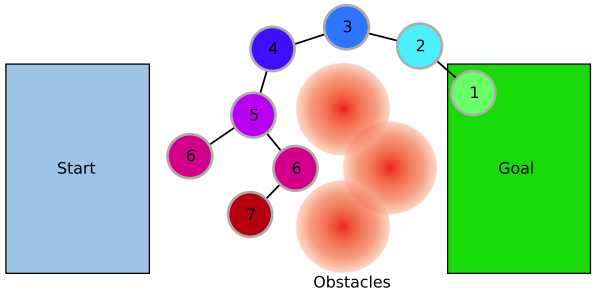

Fig. 2. A visualization of the local chain of communication applied during chain siphon behavior. The chain begins when an agent is within the region it can sense the goal and it gives itself a queue number of 1. Agents change their queue value to 1 more than the agent they are linked to. Note that 2 agents are linked to the agent with a queue value of 5, and the agent with queue value of 7 has linked with the agent closest to it.

B. Chain Siphon Algorithm Behavior

The pseudocode for the Chain Siphon Algorithm appears in Algorithm 1. The Chain Siphon Algorithm affects the swarm's behavior when at least one agent senses the goal area. The default queue value for all agents is set to the total number of agents in the swarm. An agent sensing the goal changes its queue value to 1 (lines 7-8) to indicate it is at the front of the queue, as shown in Fig. 2. Agents receive queue values from message broadcasts of neighboring agents within communication range (line 9) and record the neighbor with the lowest queue value q_{min} and shortest local distance d_{min} (lines 11-20). If an agent has not found the goal, its queue value changes to $q_i = q_{min} + 1$, up to a maximum value of n_s, and it sets $\mathbf{C_{siphon}}$ as the unit vector to the neighbor it received q_{min} from for the purposes of control (lines 21-23). Once the final value for q_i is determined, the agent broadcasts a message containing its ID with q_i to neighboring agents (line 26).

This process iterates over the swarm such that all agents within a chain of local communication become part of the queue. Agents follow the neighbors ahead of them in the queue out of any local minimum trap and eventually toward the goal.

Weighting $\mathbf{C_{siphon}}$ more than the other portions of the control equation causes each agent to follow the nearest neighbor that is lower in the queue. As w_{siphon} increases, behaviors such as adhesion between agents and goal seeking become negligible. Only control vector magnitudes that tend toward infinity will have a noticeable effect once the agent is linked. For example, in Algorithm 2 line 8 as the agent moves closer to an obstacle, d_{ij} approaches 0, causing $\mathbf{r}$ to approach infinity. Similarly, as agents draw near each other the magnitude of the Lennard-Jones potential flocking $\mathbf{r}$ tends toward negative infinity, preventing collisions.

Use of the Chain Siphon Algorithm requires that several assumptions hold. First, it assumes there are enough agents in the swarm to fill local minima, sense the goal, and form a chain

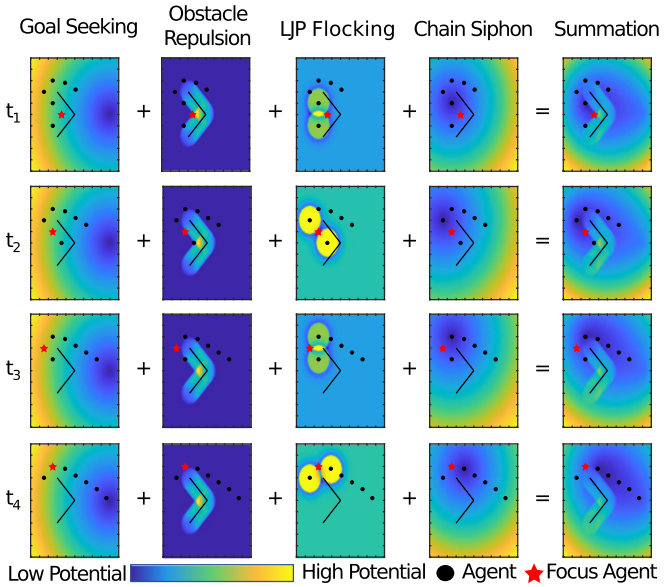

Fig. 3. Concave obstacle configurations can create a local minimum in the potential field that may trap agents. As more and more agents head toward the goal they will tend to fill in the local minimum trap until it is saturated. This will eventually cause other agents to be repelled from the area due to collision avoidance from the agents that are stuck in the local minima. Those agents continue around the obstacle until the agents' sensors detect the goal. Once the goal is detected and the chain of communication is able to link back to agents within the local minimum the chain siphon algorithm causes the swarm to exhibit the emergent behavior similar to the open channel siphon effect. The communication links propagate back through the swarm sending queue values that indicate the goal has been detected and pulling agents around the obstacles that have trapped them.

of local communication to agents stuck in the local minima. Another assumption is that the $queryNeighbors()$ function in Algorithm 1 cannot "see" through obstacles, i.e., so that agents do not link through obstacles to other agents. This assumption is reasonable if agents require line-of-sight for sensors to detect and communicate with other agents. In our simulations and experiments we enforce this assumption by increasing obstacle repulsion. Finally, our method assumes that an agent is able to sense whether or not it has reached a goal, i.e., the existence of $senseGoal()$.

C. Potential Fields

For showcasing the effect of the Chain Siphon Algorithm, the algorithm was implemented and tested using a swarm directed by goal seeking, obstacle avoiding, and flocking behaviors (see Fig. 3). The flocking behavior uses a derivative of the Lennard-Jones potential similar to Hettiarachchi *et al.* [18]. The full velocity control equation is as follows:

$$\mathbf{V_{cs}} = w_{goal} * \mathbf{C_{goal}} + w_{obstacle} * \mathbf{C_{obstacle}} + w_{LJP} * \mathbf{C_{LJP}} + w_{siphon} * \mathbf{C_{siphon}} \quad (1)$$

where the equation for the velocity vector $\mathbf{V_{cs}}$ contains scalar weighting factors w_i and control vectors $\mathbf{C_i}$ for each component of their respective behaviors. In addition to the velocity controller (Equation 1), we find that it can be useful in practice

Algorithm 2: Obstacle Repulsion

1: $a \leftarrow$ Obstacle repulsion value
2: **loop**
3: $O_i = senseObstacles()$
4: **for all** $j \in O_i$ **do**
5: $\mathbf{v_{ij}} = \mathbf{x_j} - \mathbf{x_i}$ % Rel. position vector
6: $d_{ij} = \|\mathbf{v_{ij}}\|$
7: $\mathbf{v_{ij}} = \frac{\mathbf{v_{ij}}}{\|\mathbf{v_{ij}}\|}$
8: $\mathbf{r} = -\frac{\mathbf{v_{ij}} a}{d_{ij}^2}$
9: $\mathbf{C_{obstacle}} \mathrel{+}= \mathbf{r}$
10: **return** $\mathbf{C_{obstacle}}$

to limit the maximum speed. If $\mathbf{V_{cs}}$ exceeds the specified maximum speed it is normalized and multiplied by the scalar value of the maximum speed, $\mathbf{V_{cs}} = (\frac{\mathbf{V_{cs}}}{\|\mathbf{V_{cs}}\|}) * \mathbf{v_{max}}$.

An agent's potential field in the workspace assigns the goal position as a global minimum potential while obstacles are given potentials with magnitudes approaching infinity, as seen by $\mathbf{r}$ in Algorithms 2 and 3. Interactions between agents use the Lennard-Jones potential flocking for viscoelastic fluid-like behaviors. The repulsion factor of the Lennard-Jones potential flocking acts similar to the incompressibility of the fluid, and the cohesion of the agents causes the viscous-like forces of the agents pulling each other when moving around obstacles.

D. Other Behaviors in the Control Equation

The goal seeking vector is determined through the following equation: $\mathbf{C_{goal}} = (\frac{\mathbf{x_{goal}} - \mathbf{x_i}}{\|\mathbf{x_{goal}} - \mathbf{x_i}\|}) * \mathbf{v_0}$. The resulting normalized vector is multiplied by a specified scalar initial velocity v_0 to give the vector a desired magnitude.

The obstacle repulsion algorithm used in all methods (ours and those we compare to) appears in Algorithm 2. Each agent finds the vector $\mathbf{v_{ij}}$ between the position of the agent and each obstacle sensed (line 5). The vector is normalized and multiplied by the negative of the repulsion value a and the inverse distance squared (lines 7-8). This value is added to the final vector $\mathbf{C_{obstacle}}$ (line 9).

The Lennard-Jones potential flocking is similar to the Boids model in that it exhibits cohesion and avoidance. To perform this behavior, the equation is defined as:

$$LJP_r = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right], \quad (2)$$

Hettiarachchi and Spears used (2) to find the force of the interaction between particles to drive their swarm behavior by taking the negative of its derivative,

$$F = -\left(\frac{d(LJP_r)}{dr}\right), \quad (3)$$

which becomes

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

Algorithm 3: Lennard-Jones Potential Flocking

1: $\epsilon \leftarrow$ Magnitude of attraction
2: $b \leftarrow$ Generalized parameter
3: $c \leftarrow$ Generalized parameter
4: **loop**
5: $N_i = queryNeighbors()$
6: **for all** $j \in N_i$ **do**
7: $\mathbf{v_{ij}} = \mathbf{x_j} - \mathbf{x_i}$ % Rel. position vector
8: $d_{ij} = \|\mathbf{v_{ij}}\|$
9: $\mathbf{v_{ij}} = \frac{\mathbf{v_{ij}}}{\|\mathbf{v_{ij}}\|}$
10: $\mathbf{r} = 24\epsilon\left[\frac{-26b\sigma^{12}}{d_{ij}^{14}} + \frac{7c\sigma^{6}}{d_{ij}^{8}}\right]\mathbf{v_{ij}}$
11: $\mathbf{C_{LJP}} \mathrel{+}= \mathbf{r}$
12: **return** $\mathbf{C_{LJP}}$

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

$$v = 24\epsilon\left[\frac{-26b\sigma^{12}}{r^{14}} + \frac{7c\sigma^{6}}{r^{8}}\right], \tag{5}$$

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 control equation for the Lennard-Jones potential behavior is $\mathbf{V_{LJP}} = w_{goal}\mathbf{C_{goal}} + w_{obstacle}\mathbf{C_{obstacle}} + w_{LJP}\mathbf{C_{LJP}}$.

E. Leader Heuristic Behavior (For Comparison)

We implement a leader heuristic method to compare with the Chain Siphon Algorithm. The leader heuristic method is similar to one used by Mabrouk *et al.* [14] as discussed in Section II which uses *a priori* global knowledge of local minima. Communication distances are limited to the communication ranges in the Lennard-Jones potential behavior and Chain Siphon Algorithm experiments. We allow the agents using the leader heuristic method to use global knowledge of the local minima, although the other methods examined did not require that knowledge. The leader heuristic method is similar to V_{LJP}, except in Algorithm 3 after line 10 an agent senses if a neighbor is outside of the local minima. If they are, the agent increases the attraction to that neighbor by multiplying $\mathbf{r}$ by a leader heuristic parameter e.

IV. Experiments

We conduct two sets of experiments to evaluate the Chain Siphon Algorithm and compare it with other methods. One set of experiments use real-world hardware testbed robotic blimps (Fig. 4) and run on a centralized system simulating a decentralized system (Fig. 5). These robotic experiments are run in 2-D and compare the Chain Siphon Algorithm to the Lennard-Jones potential behavior, using between 8 and 12 blimps (depending on which blimps were functional). The other set of experiments are run in a simulated environment using virtual agents (Fig. 6), which enables us to run a large number of Monte Carlo trials. The simulations compare the Chain Siphon Algorithm to the Lennard-Jones potential method and the leader heuristic method in both 2-D and 3-D environments, and using swarms with 1 to 50 agents.

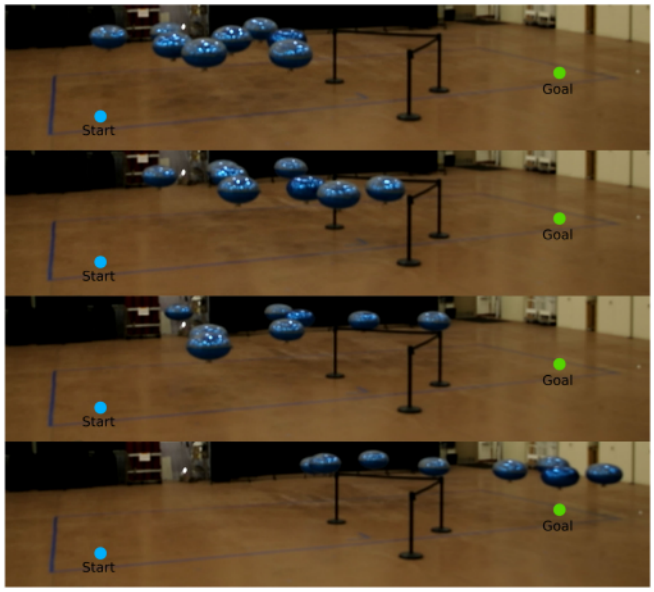

Fig. 4. A hardware experiment. GT-MABs using the chain siphon algorithm. Top to bottom show the swarm moving from start to goal.

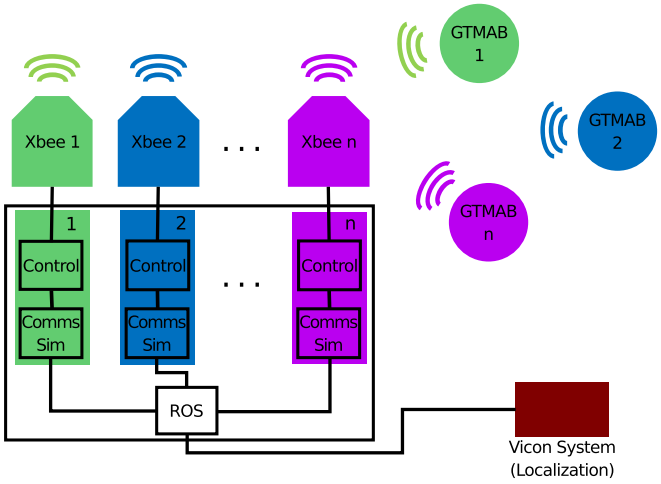

Fig. 5. A graphic of the system used in the hardware experiments.

A. Simulations

We use the SCRIMMAGE [26] platform as our 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 we set up both 2-D and 3-D simulations with a starting origin area, a goal area, and obstacles arranged in a cul-de-sac formation designed to create a local minimum. The obstacle is situated between the origin and the goal with the concave side facing the origin.

For the 2-D experiments, the obstacles are arranged in a V formation. In the 3-D experiments, five obstacles are used. Four are placed on the same plane in a square and the 5th one

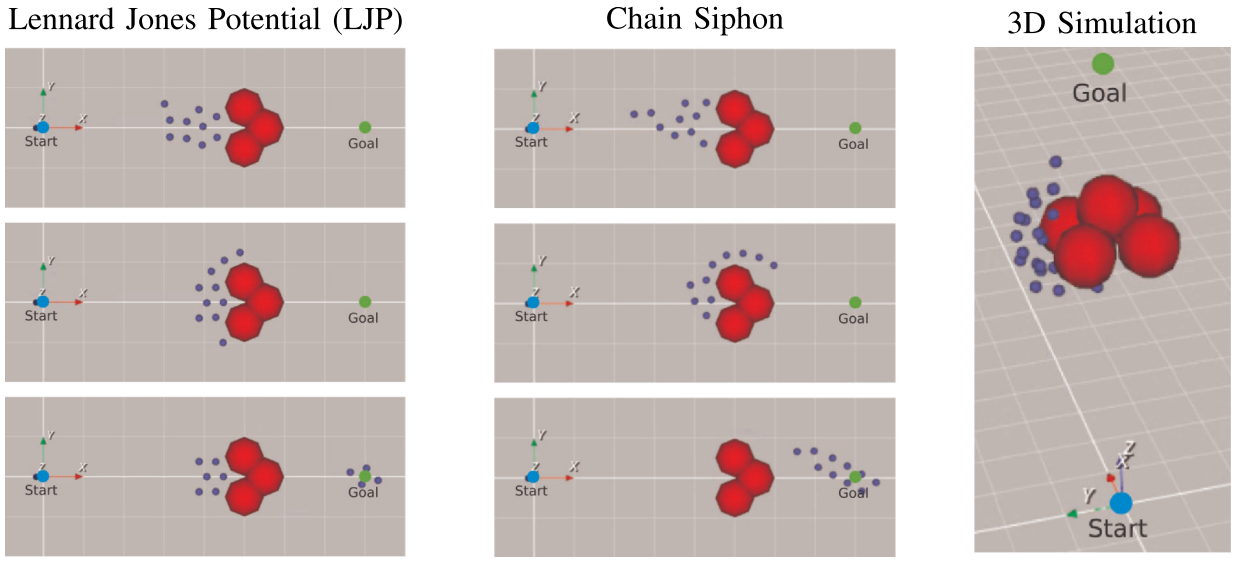

Fig. 6. SCRIMMAGE simulations showing the Lennard-Jones potential behavior, how the chain siphon algorithm pulls agents from a local minimum trap, and the 3D simulation environment.

is placed above the empty middle space between them. Both configurations are shown in Fig. 6.

Thirty Monte Carlo trials are run for each combination of algorithm, environment (2-D and 3-D), and swarm size (one to fifty agents). In each trial we randomly generate the swarm agents' starting formation near the start location. We record the number of agents that reached the goal and the number that remain trapped in local minima at the end of the trial.

B. Robotic Experiments

Hardware experiments are used to evaluate how robust the Chain Siphon Algorithm is in the physical world. The GT-MABs are a dynamically challenging platform due to their near-neutral buoyancy in ambient air. For example, a slight air flow can cause disturbances in the agents' positions.

The experiment environment is set up similar to the simulation—with an initial starting area, a goal area, and a cul-de-sac obstacle between them. The workspace is artificially constrained as seen in Fig. 4 by the markings on the floor. The Vicon position data is used for localization in the controller. ROS is used on a central computer to communicate between the Vicon system, the controller, and the Xbees that send motor commands to the blimps, as illustrated in Fig. 5.

Experiments are run with swarms containing 8-12 agents. Similar to the 2-D simulations, obstacles are represented by 3 points of 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 is not obscured by large physical obstacles.

A depiction of the swarm testbed used in our hardware experiments is presented in Fig 4. Blimps move within the workspace and position data for each blimp is provided by a Vicon motion capture system. Each blimp communicates with a ground station using a separate dedicated Xbee connection. Data exchange between blimps is limited by a communication simulator that runs on the ground station. In particular, communication is restrained such that each blimp is only allowed access to message and position data from the set of blimps within its current communication and sensor range. The movement control of each agent is computed independently on the ground station using only the message and position data currently allowed by its relative position within the swarm. All other sensing and communication happens at a local level.

V. Results

The results from the simulations can be seen in Fig. 7. For the Lennard-Jones potential behavior 2-D simulations, most of the agents become stuck when using four agents or less in the swarm. The average number of stuck agents appears to increase as the number of agents in the swarm increases. This is due to the local minimum trap area becoming saturated with agents. 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 not getting destabilized by any other agents passing nearby. 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.

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

Using a leader heuristic behavior shows for the 2-D scenario that once the swarm has grown to 5 agents there is a reduction in the number of agents getting stuck, seen in Fig. 7. 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 Fig. 7 show the same behavior, but the change in behavior initiates

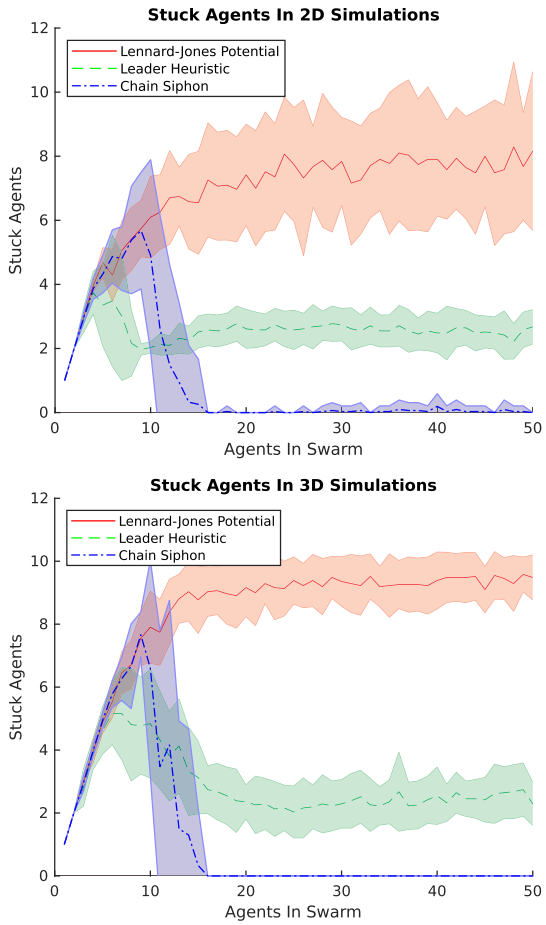

Fig. 7. The mean and the shaded area of one standard deviation of agents stuck in the local minimum trap for each type of simulation.

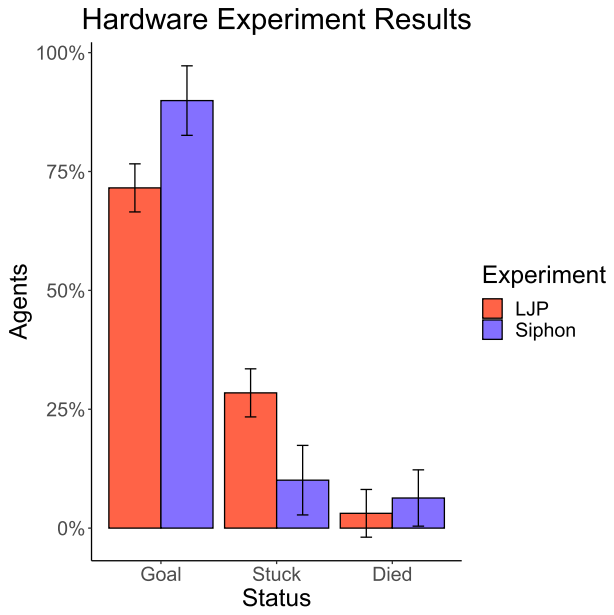

Fig. 8. The means and standard deviations of each type of blimp hardware experiment (died refers to agents with hardware malfunctions that are not related to the algorithm type). The K-S test of the data for stuck agents resulted in a p-value of 0.003.

when there are around 7 agents in the swarm, and decreases more slowly before it levels out.

The 2-D and 3-D Chain Siphon Algorithm simulations show the same behaviors as the Lennard-Jones potential behavior 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 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. The Chain Siphon Algorithm displays robustness as the swarm size increases by allowing multiple queues along different paths to form in the swarm. For the 2D environment in Fig. 6, communication chains simultaneously form above and below the obstacle to lead agents within the local minima around the obstruction.

For 3-D agents we see the siphon behavior begins to exhibit when the swarm is around 10 agents, as the average number of agents that get stuck start to decrease. This decrease continues until around 16 agents where it stabilizes at zero.

We perform ten experiments for both the Lennard-Jones potential behavior and Chain Siphon Algorithm, and the results are shown in Fig. 8. The average number of agents stuck during the Lennard-Jones potential behavior experiments is 2.7, 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 Algorithm 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.

A K-S test of the two sets of data for stuck agents gives a p-value of 0.003 (where anything < 0.05 is considered to be statistically significantly).

VI. Conclusion

In this paper we have presented the Chain Siphon Algorithm: a novel algorithm inspired by the open channel siphon effect that assists swarms navigating environments with local minima. The algorithm is designed for implementation onto distributed systems by using local chains of communications to relay information throughout the system. Results from hardware experiments and simulations have shown statistically significant mitigation of local minima traps compared to the Lennard-Jones potential behavior and/or a leader heuristic method.

The results from both the 2-D and 3-D simulations (Fig. 7) show a decrease in the number of agents stuck in a local minimum once the number of agents in the swarm is large enough to establish a chain of communication. As the swarm becomes

larger it creates the chain of communication more consistently and shows less agents become stuck compared to the other methods. The hardware experiment results also show using the Chain Siphon algorithm decreases the number of agents stuck in local minima to a significant degree.

APPENDIX A
PARAMETER SELECTION

Parameters were determined by manually selecting and testing values to find the high and low limits of values that show desired behaviors. We perform a manual parameter sweep to determine the final parameter values. Desired behaviors of each component of the control equation are established and parameters are chosen that best exhibited those behaviors.

The parameters within the simulation environment are unitless, although simulated range values are analogous to distances and speed parameters are analogous to velocities. We list them here to provide reference to the relative scale each parameter is given relative to other parameters and to the parameters of the hardware experiments. The initial speed is set to 10. The weights for the control equation components are set to 1, except for the chain siphon component which is set to 75. The obstacle range is designated as 30 and obstacle repulsion is set to 1. The neighbor range is assigned 5, ϵ is 0.25, collision range is set to 2, and maximum speed is limited to 30.

In the hardware experiments, the maximum speed the motors enable the blimp to travel is approximately 0.4 m/s. The controller limits the motor speed to 75% of the maximum speed while the initial speed is 50% of the maximum speed. The range an agent senses obstacles is 1.5 m, and the obstacle repulsion value is set to 10. The range for sensing neighbors is 2.25 m, ϵ 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 behavior weight is set to 0.05. During Lennard-Jones potential behavior experiments the Chain Siphon Algorithm weight is set to 0, but during Chain Siphon Algorithm experiments it is set to 20.0. A value of 50 is given to a control behavior that keeps the blimps within the designated workspace.

REFERENCES

[1] J. Gibson, T. Schuler, L. McGuire, D. M. Lofaro, and D. Sofge, "Swarm and multi-agent time-based a* path planning for lighter-than-air systems," *Unmanned Syst.*, vol. 8, no. 3, pp. 253–260, 2020. [Online]. Available: https://doi.org/10.1142/S2301385020500181

[2] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.*, vol. 5, no. 1, pp. 90–98, 1986. [Online]. Available: https://doi.org/10.1177/027836498600500106

[3] L. Barnes, W. Alvis, M. Fields, K. Valavanis, and W. Moreno, "Swarm formation control with potential fields formed by bivariate normal functions," in *Proc. 14th Mediterranean Conf. Control Automat.*, 2006, pp. 1–7.

[4] L. Barnes, M. Fields, and K. Valavanis, "Unmanned ground vehicle swarm formation control using potential fields," in *Proc. Mediterranean Conf. Control Automat.*, 2007, pp. 1–8.

[5] Q. Chen, S. Veres, Y. Wang, and Y. Meng, "Virtual spring-damper mesh-based formation control for spacecraft swarms in potential fields," *J. Guid., Control, Dyn.*, vol. 38, pp. 539–546, 2015.

[6] C. Bentes and O. Saotome, "Dynamic swarm formation with potential fields and a* path planning in 3D environment," in *Proc. Braz. Robot. Symp. Latin Amer. Robot. Symp.*, 2012, pp. 74–78.

[7] 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 *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 2347–2354.

[8] F. Yegenoglu and H. Stephanou, "Collision-free path planning for multirobot systems," in *Proc. IEEE Int. Symp. Intell. Control*, 1989, pp. 537–542.

[9] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. ICNN'95 - Int. Conf. Neural Netw.*, 1995, vol. 4, pp. 1942–1948.

[10] R. Palm and A. Bouguerra, "Particle swarm optimization of potential fields for obstacle avoidance," in *Proc. RARM Scient. Coop. Intern. Conf. Elect. Electr. Eng.*, Istanbul, Turkey, pp. 117–123, Sep. 2013.

[11] D. Zhou, Z. Wang, and M. Schwager, "Agile coordination and assistive collision avoidance for quadrotor swarms using virtual structures," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 916–923, Aug. 2018.

[12] C. Kownacki and L. Ambroziak, "Local and asymmetrical potential field approach to leader tracking problem in rigid formations of fixed-wing UAVs," *Aerosp. Sci. Technol.*, vol. 68, pp. 465–474, 2017.

[13] J. H.Reif and H. Wang, "Social potential fields: A distributed behavioral control for autonomous robots," *Robot. Auton. Syst.*, vol. 27, no. 3, pp. 171–194, 1999. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889099000044

[14] M. H. Mabrouk and C. R. McInnes, "Swarm robot social potential fields with internal agent dynamics," in *Proc. Int. Conf. Aerosp. Sci. Aviation Technol. (ASAT12), ROB-02:1-14*, Cairo, Egypt, vol. 12, 2007.

[15] H. E. Espitia and J. I. Sofrony, "Path planning of mobile robots using potential fields and swarms of Brownian particles," in *Proc. IEEE Congr. Evol. Comput.*, 2011, pp. 123–129.

[16] F. Matoui, B. Boussaid, and A. Mohamed Naceur, "Local minimum solution for the potential field method in multiple robot motion planning task," in *Proc. Int. Conf. Sci. Techn. Autom. Control Comput. Eng.*, 2015, pp. 452–457.

[17] S. Hettiarachchi and W. M. Spears, "Distributed adaptive swarm for obstacle avoidance," *Int. J. Intell. Comput. Cybern.*, vol. 2, no. 4, pp. 644–671, 2009. [Online]. Available: https://doi.org/10.1108/17563780911005827

[18] S. Hettiarachchi, W. Spears, D. Green, and W. Kerr, "Distributed agent evolution with dynamic adaptation to local unexpected scenarios," in *Proc. Workshop Radical Agent Concepts*, 2005, vol. 3825, pp. 245–256.

[19] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 25–34, Aug. 1987. [Online]. Available: http://doi.acm.org/10.1145/37402.37406

[20] J. E. Jones, "On the determination of molecular fields. II. From the equation of state of a gas," in *Proc. Roy. Soc. London Ser. A*, 1924, vol. 106, no. 738, pp. 463–477.

[21] H. G. Tanner, A. Jadbabaie, and G. J. Pappas, "Stable flocking of mobile agents, part I: Fixed topology," in *Proc. 42nd IEEE Int. Conf. Decis. Control* (IEEE Cat. No. 03CH37475), 2003, vol. 2, pp. 2010–2015 .

[22] H. G. Tanner, A. Jadbabaie, and G. J. Pappas, "Stable flocking of mobile agents part II: dynamic topology," in *Proc. 42nd IEEE Int. Conf. Decis. Control* (IEEE Cat. No.03CH37475), 2003, vol. 2, pp. 2016–2021 .

[23] H. G. Tanner, A. Jadbabaie, and G. J. Pappas, "Coordination of multiple autonomous vehicles," in *Proc. IEEE Mediterranian Conf. Control Automat.*, 2003, pp. 869–876.

[24] S. Cho *et al.*, "Autopilot design for a class of miniature autonomous blimps," in *Proc. IEEE Conf. Control Technol. Appl.*2017, pp. 841–846.

[25] Q. Tao, J. Cha, M. Hou, and F. Zhang, "Parameter identification of blimp dynamics through swinging motion," in *Proc. Int. Conf. Control, Automat., Robot. Vis.*, 2018, pp. 1186–1191.

[26] K. DeMarco, E. Squires, M. Day, and C. Pippin, "Simulating collaborative robots in a massive multi-agent game environment (SCRIMMAGE)," in *Proc. Int. Symp. Distrib. Auton. Robotic Syst.*, 2018, pp. 283–297.