Control of Swarms with Multiple Leader Agents

Phillip Walker, Saman Amirpour Amraii, and Michael Lewis
School of Information Sciences
University of Pittsburgh
Pittsburgh, PA 15213, USA
pmw19@pitt.edu, samirpour@acm.org, ml@sis.pitt.edu

Nilanjan Chakraborty and Katia Sycara
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
nilanjan@cs.cmu.edu, katia@cs.cmu.edu

Abstract— **The study of human control of robotic swarms involves designing interfaces and algorithms for allowing a human operator to influence a swarm of robots. One of the main difficulties, however, is determining how to most effectively influence the swarm after it has been deployed. Past work has focused on influencing the swarm via statically selected leaders—swarm members that the operator directly controls. This paper investigates the use of a small subset of the swarm as leaders that are *dynamically selected* during the scenario execution and are directly controlled by the human operator to guide the rest of the swarm, which is operating under a flocking-style algorithm. The goal of the operator in this study is to move the swarm to goal regions that arise dynamically in the environment. We experimentally investigated three different aspects of dynamic leader-based swarm control and their interactions: leader density (in terms of guaranteed hops to a leader), sensing error, and method of information propagation from leaders to the rest of the swarm. Our results show that, while there was a large drop in the number of goals reached when moving from a 1-hop to a 2-hop guarantee, the difference between a 2-hop, 3-hop, and 4-hop guarantee was not statistically significant. Furthermore, we found that sensing error impacted the *flooding* information-propagation method more than the *consensus* method conditions, and caused participants more trouble the lower the density of leaders.**

I. INTRODUCTION

One of the primary benefits of robotic swarms is that they make use of robust scalable algorithms to coordinate tens or hundreds of robots toward achieving some common goal. Examples highlighting swarm scalability and robustness include coverage of an environment [1], path-planning [2], self-assembly [3], and network routing optimization [4]. Because of their scalability and reliance on emergent coordination, swarms are often robust to the failure of individual members, and thus can make use of robots that are cheaper to produce. Despite these advantages of robotic swarms, there are many challenges arising due to the unpredictability of swarm algorithms. While many of these algorithms have proven outcomes and guarantees [5], the proofs require assumptions that are rarely found in the real world, such as obstacle-free environments. Therefore, there is a need for a human operator to oversee the swarm operation and give both goal-directed input and correct unforeseen errors in the swarm's operation.

This research has been sponsored in part by NSF Grant CNS-1329762 and ONR Grant N0001409-10680.

Controlling a swarm through direct teleoperation of individual robots is challenging because, in addition to the direct control over the robots assigned to them, an operator of a swarm would also have to deal with the local interactions with other robots, thus making the control of a swarm super-linear in the number of robots [6]. Swarm algorithms are therefore designed to handle the local interactions autonomously, which both alleviates the need for a human operator to handle them and allows for emergent, global behaviors to arise. The emergent behavior of swarms, on the other hand makes human control difficult—especially in complex obstacle-filled environments.

There are different models of robotic swarms, such as broadcasting commands to all robots simultaneously [7], and treating the swarm as a spatial computer [8]. In this paper, we study human control of bio-inspired swarms—namely swarms that operate via simple local control laws. Bio-inspired swarm control borrows significantly from collective animal behavior, such as schooling fish and flocking birds. In fact, one of the original implementations of swarming algorithms comes from Reynolds' 1987 paper on creating a virtual flock of birds for computer animation purposes [9]. Iain Couzin has attempted to model the local rules governing flocking behavior in animals. He designates different zones whereby the animals obey different rules governing their interactions with neighbors depending on what zone those neighbors are in [10]. More recently, Couzin has investigated how leaders within such groups can influence the overall movement of the swarm without directly telling the followers what to do [11]. Here, the authors showed that only a small percentage of the swarm need be knowledgeable for this to work, and that this percentage shrinks as the swarm grows larger, further lowering the control requirements for any human-swarm system using this model.

Influencing bio-inspired swarms via direct control of one or a subset of robots that are termed leaders has been studied in the literature. Leader-based methods of swarm control are attractive because they are well-suited for the conditions under which swarms will likely operate in the real world.

Because of the need to make individual swarm members as cost-effective as possible, and last as long as possible, it may not be practical to have each robot communicate directly with the human operator at all times, as long range communication requires a significant amount of energy. Therefore, selecting one or a subset of the swarm as leaders is advantageous.

In the reported work to-date, leaders are statically selected. In contrast, in this paper we employ a dynamic scheme for leader selection. In other words, leaders change during the execution of a mission, based on criteria such as communication range. In practice, dynamic selection could be preferable to static selection schemes for a number of reasons. For example, it would ensure that the power requirements of long-range communication are distributed throughout the swarm as it moves, rather than having a small set of robots always using long-range communication to send and receive information from the operator.

In this paper, we conduct a user study of swarm control with dynamically selected leaders. To our knowledge this is the first study to employ such a scheme. In particular, we studied the effect of (a) leader density, (b) sensing error, and (c) method of information propagation on system performance and ease of human control of the swarm in an information foraging study with obstacles and dynamically arising goals regions. Our results show that it is possible for humans to control a simulated swarm of robots in such a scenario even with significant error, although the method of information propagation is important when determining the effect sensing error will have. Furthermore, we show that different numbers of leaders can be selected effectively, thus tailoring the selection to the needs of particular applications.

A. Overview and Hypotheses

In previous work [12], we looked at two different methods of propagating influence, in the form of a numerical value, from a single leader to the rest of a stationary network of nodes, and investigated what effect the placement of that leader in the communication graph had on convergence time. These two methods were *flooding* and *consensus* propagation. Flooding involves directly passing the goal value to neighboring robots, who in turn pass it to their neighbors, and so on—thus “flooding” the network. Consensus involves each robot averaging the values of all its neighbors, regardless of their leader status. A follow-up paper investigated these propagation methods on a simulated swarm of robots with a single leader, and showed the benefits of each depending on the operating scenario [13].

The current paper extends the work of [13] to multiple dynamically selected leaders and to obstacle-filled environments. We use a modified version of the Random Competition Clustering (RCC) algorithm [14], used to select cluster heads for wireless sensor networks (WSNs), to dynamically select leaders. As a measure of density we used the notion of *n-hop guarantee*, used in algorithms of dynamic cluster head selection in wireless networks. An n-hop guarantee means that every robot is guaranteed to be at most n hops away from a leader along the inter-robot communication or sensing graph. In this study, we investigated the effect of 1-hop, 2-hop, 3-hop, and 4-hop guarantees on system performance. In each of these hop-guarantee conditions, the operator directly controlled the corresponding dynamically-selected leaders that resulted from the execution of the leader selection algorithm (see next section).

Our hypothesis is that, as the number of hops increases (and by extension, the number of leaders decreases), the swarm will be harder to control, and thus performance will decrease. We also hypothesize that error will have an increasingly negative effect the lower the density of leaders, and that the *flooding* propagation method will suffer more from sensing error that the *consensus* method.

Overall, we hope to demonstrate that using dynamic leaders is an effective way to influence a swarm, even when communication between the human and the swarm is limited and there are obstacles in the environment. We also wish to investigate what effect changing the density of leaders, through different n-hop-to-leader guarantees, has on the ability of the human to influence the swarm—perhaps giving us a rough performance curve that could be further investigated in future studies to determine the optimal number of leaders for a given swarm setup. In section II, we introduce the experiment setup and describe the swarm algorithms and the nature of the task. In section III we present and discuss the results of the study. Finally, in section IV we conclude and discuss possibilities for follow-up research.

II. TASK DESCRIPTION

Our study investigates the ability of human operators to control a flocking swarm of robots in an obstacle-filled environment by teleoperating multiple dynamically-selected leaders via a continuous velocity (i.e. heading and speed) command. The main task for the users is to survey a given area by guiding the swarm to goal regions, which appear dynamically in the environment. A goal region appears at a random position only after another goal region has been visited by the swarm. Thus, the number of goal regions visited by the operator is a natural measure of his or her ability to control the robots. As stated earlier, we investigate the differences between two methods of propagating the heading and speed command from the leader to the rest of the swarm. In addition, we investigate the effect of sensing error of the robots on the ability of the operator to control the swarm. There were 16 different possible conditions, shown in Table I. The experiment was a between-subjects design, with each participant completing one condition.

A. The Environment and Robots

We use an obstacle-filled 100x100 meter environment with 200 robots, each 60x60 centimeters, for the study. The robot controllers and user interface are implemented in a Java applet deployed on a web page for completion by workers from the Amazon Mechanical Turk marketplace.

Each robot is equipped with a simulated neighbor sensor to determine the speed and heading of neighboring robots within 4 meters. In the conditions with sensing error, noise

prop. method	error model	hops	condition	N
consensus	no error	1	1	11
consensus	error	1	2	15
consensus	no error	2	3	15
consensus	error	2	4	12
consensus	no error	3	5	15
consensus	error	3	6	7
consensus	no error	4	7	10
consensus	error	4	8	10
flooding	no error	1	9	12
flooding	error	1	10	14
flooding	no error	2	11	13
flooding	error	2	12	13
flooding	no error	3	13	13
flooding	error	3	14	10
flooding	no error	4	15	13
flooding	error	4	16	9
			Total	192

TABLE I
TABLE SHOWING THE 16 DIFFERENT CONDITIONS EMPLOYED IN THE USER STUDY.

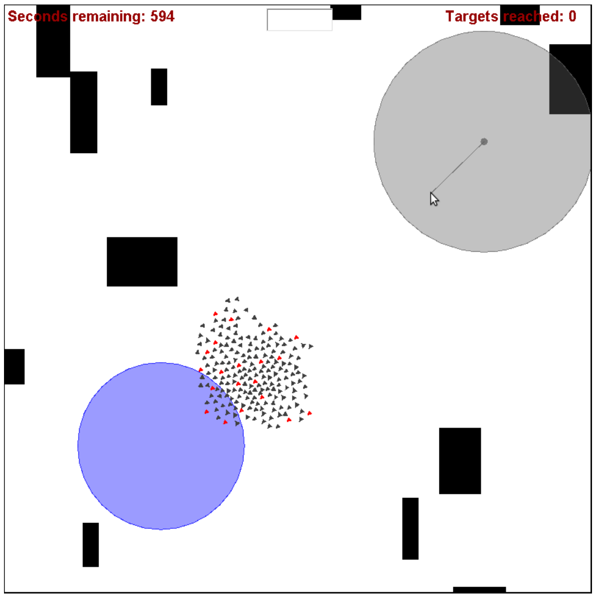

Fig. 1. The user interface during a sample experiment session. The operator is using their mouse to bring up the virtual joystick (top right) to guide the swarm of robots (middle) to the blue goal region (bottom left). The leader robots are shown in red. Free space in the environment was white, while obstacles were shown in black.

variables e_x and e_y are added to the ground truth coordinates x and y of each sensed robot, and updated each time the robot moved. When updating, two randomly sampled amounts from the uniform distribution $[-5.0, 5.0]$ centimeters are added to e_x and e_y respectively; however the absolute values of both e_x and e_y are not allowed to exceed 0.5 meters. Similarly, an error value e_θ is added to the sensed robot's heading. This error value changes by an amount sampled from the uniform distribution $[-5.0, 5.0]$ degrees and added to the sensed robot's heading each time it moves. Furthermore, the absolute value of e_θ can not exceed 45 degrees.

In each of the four conditions, the swarm of robots is initialized randomly in a 10x10 meter box, centered around the origin of the environment, each with a random starting orientation. There are no collision models defined between robots, meaning robots can freely overlap each other. However, there does exist a collision model between the robots and the obstacles, such that if each robot comes into contact with an obstacle, it is considered stuck for the remainder of the experiment. Contact occurs when any part of the environment within 30 centimeters (half the robot diameter) of the robot center is occupied by an obstacle.

B. Human Influence and Control Algorithm

The main goal of the participant is to steer the swarm to the goal region shown as a blue circle in the environment (Figure 1). Once 50% of the swarm (100 robots) are in the goal region simultaneously, the goal will move to a new random position. We used 50% as the threshold because it was large enough to prevent stray robots or small groups from triggering a new goal, but small enough that a swarm with a large diameter (or many lost robots) could still fit enough robots in the region.

To influence the swarm, the operator could click the left mouse button to open up a virtual joystick with which to steer the leader robots by dragging the mouse to form a line on the joystick (Figure 1). This joystick allows the participants to set both the heading and speed of all the leaders simultaneously. The goal speed sent to the leaders is calculated as

$$s = 2 * \frac{min(l,r)}{r} \tag{1}$$

where l is the length of the line in pixels drawn by the user, r is the radius of the joystick circle (300 pixels in this study), and s is the speed in meters per second. The global heading goal sent to the leaders is simply the heading of the line drawn by the user. The other robots move according to local control laws, hereafter called the *align*, *attract*, and *repel* laws, which are explained below.

1) Alignment and Speed: The alignment vector is determined by the propagation method (flooding or consensus). For the *flooding* propagation condition, each robot finds its neighbor with the lowest state, as defined by the leader selection algorithm (see Section II-C). The robot will set its velocity and its alignment vector, $\langle x_a, y_a \rangle$, to match the velocity and heading of that neighbor in the global coordinate frame. In the event that two neighbors share the lowest state value, the one with the lower ID will be chosen. Note that, in the conditions with sensing error, the neighbor heading being matched may be inaccurate, as it is based on the heading returned by the simulated neighbor sensor.

For the *consensus* propagation method, each robot averages the speed and heading of every neighbor it can sense, and then sets its speed and alignment vector $\langle x_a, y_a \rangle$ to that average speed and heading (again in the global coordinate frame). There is no special status given to the states of neighbors.

2) Attraction and Repulsion: In addition to sensing neighbors' motion vectors, the robots also sense neighbors' positions to maintain swarm cohesion and avoid inter-robot collisions through attraction and repulsion laws. The attraction vector $\langle x_c, y_c \rangle$ is determined by summing the relative headings to all neighboring robots that are outside the attraction minimum range of 2.0 meters. The repulsion vector $\langle x_r, y_r \rangle$ is determined by summing the negative relative headings to all neighboring robots that are inside the 2-meter threshold. All three of the vectors (align, attract, and repel) are then normalized and summed to give the instantaneous motion vector, according to the equation below.

$$\langle x_g, y_g \rangle = \langle x_a, y_a \rangle + \langle x_c, y_c \rangle + \langle x_r, y_r \rangle \tag{2}$$

C. Leader Selection Algorithm

In order to distribute the leaders as evenly as possible across the swarm while still restricting communication to neighbors only, we used a modified version of the Random Competition Clustering (RCC) algorithm for selecting cluster heads in WSNs [14]. This algorithm runs on each robot, and takes as input the period of an asynchronous leader update timer t, and an integer value x, where x is maximum guaranteed number of hops to a leader for the given experiment condition. For our experiment, we used a value of $t = 1 + r$ seconds across all conditions, where r is a random value sampled from the uniform distribution $[-0.1, 0.1]$. Psuedocode for the following algorithm can be found in Algorithm 1.

When the algorithm begins, the robot initializes its state variables $s = x$ (Line 2), and then start two timers. First, $T_{broadcast}$ (Line 3), which goes off with a period of 50Hz and triggers a $broadcast(id, s)$ event (Lines 6-10). The second, $T_{leader} = t$ (Line 4), represents the current time remaining until the robot checks its leader state (Lines 11-19). When this leader timer expires, the timer is restarted at t seconds (Lines 17-18), and any robot that is currently a leader, but with fewer than three neighbors, will lose its leader status and reinitialize its state $s = x$ (Lines 12-13). If the robot is not a leader when T_{leader} expires, it will become one if it has at least three neighbors in its set of neighbors, N. (Lines 14-16). This was implemented to let lone robots (or pairs) to wander, allowing them to be "picked up" again by the swarm if the operator navigated near the lone robot(s). Without this, the robot(s) would be stuck following along with operator commands in formation with, yet out of range of, the main swarm, making them useless.

If not responding to a timer event, the robot is constantly reacting to new neighbor messages it receives. If the robot receives a state message of $s_i < s$ from any neighbor before its timer expires, it will reset $T_{leader} = t$, and set its current state value to $s = s_i + 1$ (Lines 21-24). If, however, a robot is currently a leader, and receives a message from a robot with a lower ID and a state $s_i < x - 1$, then this robot will defer leadership to the other robot's leader, set its own state to $s_i + 1$, and restart the leader timer (Lines 25-29).

The main modification from the standard RCC algorithm is that of the state variable, giving the ability to set a guaranteed number of hops from any given leader. In the

Algorithm 1 *modified_RCC (int t, int x)*, where t is the interval of the leader timer described in Section II-C and x is the maximum hops any robot can be from a leader.

```
1: Define $id$ = robot's ID
2: State $s = x$
3: Start Timer $T_{broadcast} = 0.02$
4: Start Timer $T_{leader} = t$
5: while $true$ do
6:     if $T_{broadcast} = 0$ then
7:         $broadcast(id, s)$
8:         $T_{broadcast} = 0.02$
9:         Start Timer $T_{broadcast}$
10:    end if
11:    if $T_{leader} = 0$ then
12:        if $s = 0$ and $|N| < 3$ then
13:            $s = x$
14:        else if $|N| \geq 3$ then
15:            $s = 0$
16:        end if
17:        $T_{leader} = t$
18:        Start Timer $T_{leader}$
19:    end if
20:    for $m_i \in M$, where $M$ is the set of neighbor messages do
21:        if state $s_i \in m_i < s$ then
22:            $s = s_i + 1$
23:            $T_{leader} = t$
24:            Start Timer $T_{leader}$
25:        else if state $s_i \in m_i < x - 1$ and $s = 0$ and $id_i \in m_i > id$ then
26:            $s = x$
27:            $T_{leader} = t$
28:            Start Timer $T_{leader}$
29:        end if
30:    end for
31: end while
```

RCC algorithm, each robot starts with a timer, and then declares itself a leader if the time expires before it receives a message from any neighbor indicating that they are a leader. This guarantees that each robot is connected to a leader upon completion. While the RCC algorithm was initially used for a one-time selecting of cluster heads in stationary wireless sensor networks, it is also well-suited for our problem of dynamically selecting leaders in a swarm of robots. Therefore, we made the second modification of adding the leader timer, T_{leader} to switch a robot's leader state.

D. Conditions and Experiment

Participants were gathered from the Amazon Mechanical Turk marketplace. Each participant completed one condition of the study, and was paid \$0.50 plus \$0.25 per goal reached. Conditions incremented after each participant, repeating 1-16 (see Table I). In all of the conditions, the goal of the operator was to influence the swarm to move toward the goal region through teleoperation of the leaders (Figure 1). Once 100 or more of the 200 robots reached the goal region, it would immediately move to a new random position free of obstacle overlap.

Each participant was first guided to an instructions screen, which explained the participant's goal during the study, and how to use the interface controls. Then, each participant was given up to five minutes time to train with the interface, on their condition, to understand the controls and properties of the experiment unique to that condition. Finally they began the main 10 minute experiment section where they attempted to reach as many goal regions as possible. The conditions had different, random sets of obstacles in the environment (see Figure 1 for an example). There were 155 participants in total (see Table I for the distribution across conditions).

III. RESULTS AND DISCUSSION

All statistical significance in this section was established through one-way ANOVAs or pairwise t-tests. The main measure of success for participants is the number of goal regions reached over the course of the 10 minutes main experiment session. Results below are given with mean values M, F-statistics $F()$, and p-values p.

A. *Operator Performance*

For the main results, we first found that propagation type significantly impacted the operators' ability to reach goals, with the *flooding* method allowing more goals to be found on average ($M = 10.4$) than the *consensus* method ($M = 3.96$, $F(1,190) = 49.77$, $p < .001$). Similar differences were found between the error and no error conditions, although the results were marginally significant. Participants in the no error conditions found more goals ($M = 8.21$) than the error conditions ($M = 6.09$, $F(1,190) = 4.33$, $p = .039$). For the four different hop guarantees, we found that participants in the 1-hop condition reached significantly more goals ($M = 13.8$) than the 2-hop ($M = 5.09$, $t = 6.74$, $p < .001$), 3-hop ($M = 4.82$, $t = 6.62$, $p < .001$), and 4-hop ($M = 4.31$, $t = 7.42$, $p < .001$) conditions. However, there were no significant differences between any of the other hop guarantees. These results confirmed our hypotheses, and suggest that, in general, direct propagation through the *flooding* method is superior to *consensus*, and that error would affect performance overall. Furthermore, it suggests that the influence of leaders degrades sub-linearly as the leader density decreases. Moving from 1-hop to 2-hops gives significant performance degradation, yet this quickly disappears as you move to lower leader densities.

Analysis of the interaction between sensing error and propagation type confirmed our earlier hypotheses. Participants in the *flooding* conditions with sensing error reached significantly fewer goals ($M = 8.04$) than those in the *flooding* conditions without sensing error ($M = 12.5$, $F(1,95) = 9.02$, $p = .003$). However, participants in the *consensus* conditions showed no difference between those with sensing error ($M = 4.46$) and those without ($M = 4.22$, $F(1,93) = 0.029$, $p = .864$, see Figure 2). This shows that the domination of the *flooding* method may not hold for situations with high error. We also found an interesting interaction between error and leader density, with error significantly affecting those participants in the 3-hop and 4-hop density conditions, but not the 1-hop and 2-hop conditions (see Figure 3).

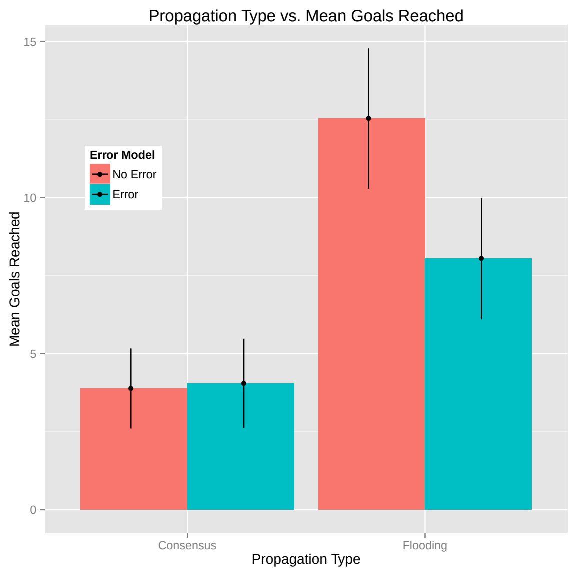

Fig. 2. The number of goals reached for each propagation type, separated by error model.

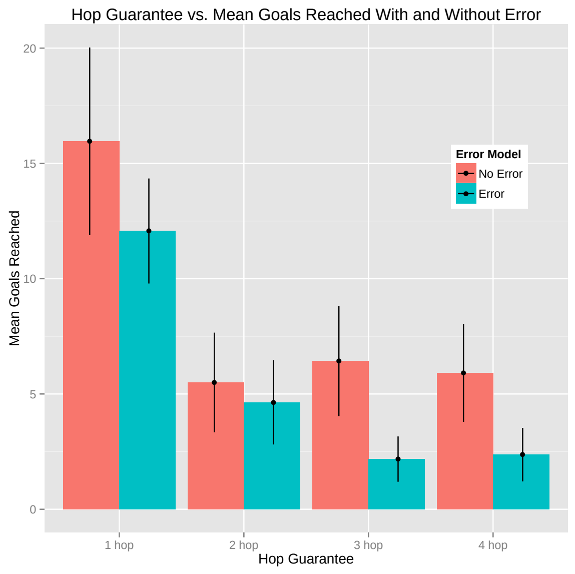

Fig. 3. The number of goals reached by each hop guarantee, separated by error model.

Further analysis of the hops conditions reveals that, although there were significant differences in the average number of leaders between each of the different densities, the 1-hop conditions featured almost twice as many leaders ($M = 35.3$) as any of the others. The difference between any two other densities, while significant, was significantly smaller (see Figure 4).

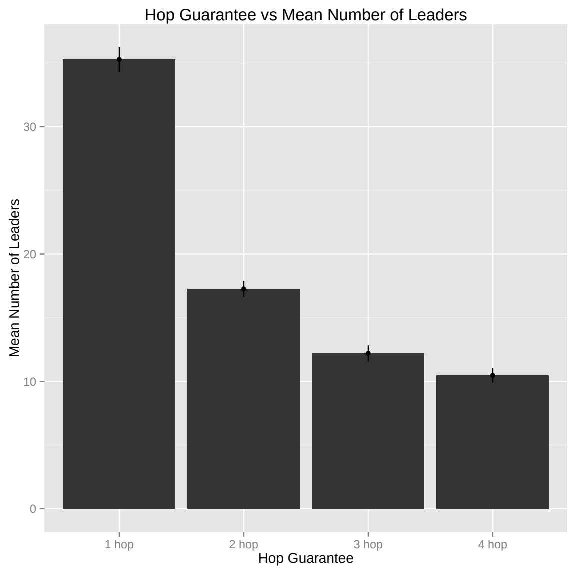

Fig. 4. The average number of leaders during an experimental session across the aggregated 1-, 2-, 3-, and 4-hop leader selection conditions.

B. Operator Behaviors

We also investigated the effect of operator behaviors on the number of goals found. For the following results, participants that reached no goals during their 10-minute trial were removed. Overall results showed that the faster the participants moved the swarm (as measured by the average speed of the swarm centroid) the fewer the number of goals reached ($p < .001$, $r^2 = .082$). However, further investigation showed that this was due to the *consensus* conditions only ($p < .001$, $r^2 = .234$). The *flooding* conditions showed no such correlation ($p = .283$, $r^2 = .002$). Swarm centroid speed also positively correlated with more robots disconnected from the largest connected group of robots ($p < .001$, $r^2 = .548$). This correlation was stronger with the *consensus* conditions ($p < .001$, $r^2 = .649$) than with the *flooding* conditions ($p < .001$, $r^2 = .200$, see Figure 5).

On a related note, the average speed of the swarm also impacted the overall swarm diameter (the furthest distance between two robots in the largest connected group of the swarm). However, the outcome was different depending on the propagation type. For the *flooding* conditions, participants who moved the swarm faster saw a larger diameter ($p = .001$, $r^2 = .093$), yet those in the *consensus* conditions who moved the swarm faster saw a smaller diameter ($p < .001$, $r^2 = .326$, see Figure 6). Notice that this effect was larger in the *consensus* than in the *flooding* conditions.

We believe a smaller diameter swarm for this study is beneficial, because it can avoid obstacles more easily and is easier to contain within a goal region. However, to reach this operators are forced to move their swarm slower than they would like. It may initially seem that the negative correlation

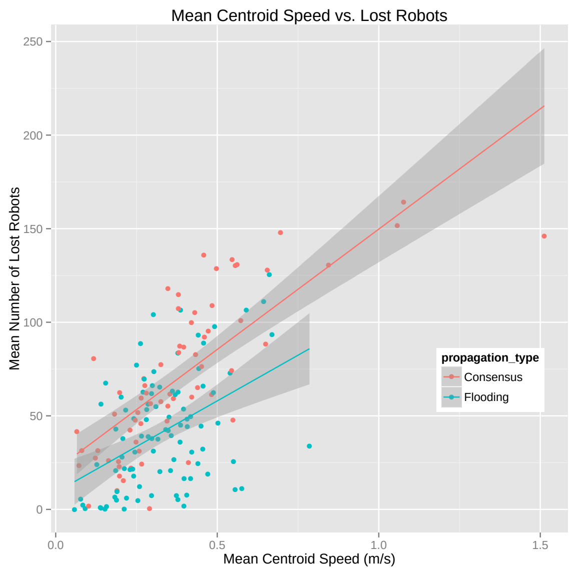

Fig. 5. Figure showing the correlation between the mean centroid speed and the mean number of lost robots during the experiment. Participants that reached no targets were removed.

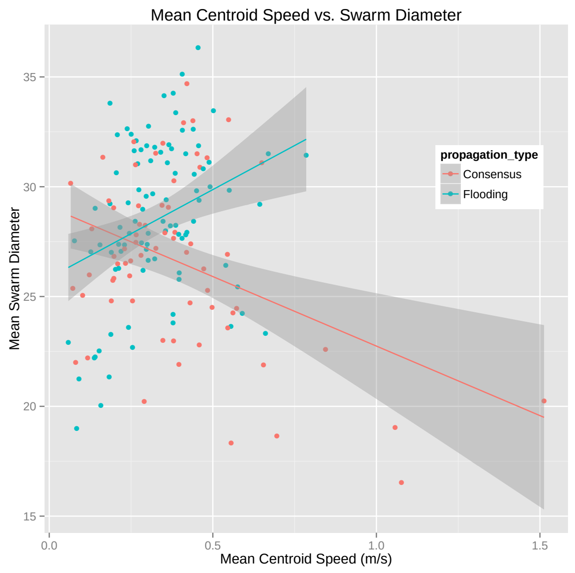

Fig. 6. The effect of average swarm speed on the average diameter of the swarm for both propagation types. Participants that reached no targets were removed.

found between centroid speed and diameter in the *consensus* conditions indicates otherwise. As the reader can see from Figure 5, however, those moving at the highest speed in the *consensus* condition lost a prohibitively high number of robots. Therefore, operators in the *flooding* conditions were better served by the design, because they suffered fewer lost robots overall—instead the overall diameter of the swarm grew—and thus saw some benefit from moving slightly faster. In other words, participants in the *flooding* conditions who moved their swarms faster saw increased diameters of their swarm, whereas those in the *consensus* conditions lost robots.

IV. CONCLUSIONS AND FUTURE WORKS

Overall, this paper demonstrates that controlling a swarm by dynamically selecting multiple leaders to act as intermediaries between the operator and the rest of the swarm is a practical method of swarm control for an information-foraging task. We showed that, under most circumstances, direct information propagation via the *flooding* method is superior to the indirect *consensus* method; however, with significant sensing error this may not always be the case. Finally, we showed that lowering the leader density from one to more than one hop produced a sharp decrease in performance but further reductions had little impact. The effects of error on performance, however, were noticeable between 2 and 3 hops where errorless performance was unaffected.

In the future, we hope to develop a hybrid method to provide the fast propagation that the *flooding* method provides, with the robustness to error given by the *consensus* method. Furthermore, we would like to look at how to handle environments with more complex obstacles, such as long hallways or small doorways, where leaders on different sides of the swarm may need to receive different commands than others, due to the relative layout of obstacles (e.g., moving the swarm around a corner). Overall, this study shows that leader-based swarm control is viable, and can be adapted successfully to different mission conditions.

REFERENCES

[1] N. Correll and A. Martinoli, "Robust distributed coverage using a swarm of miniature robots," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 379–384.

[2] G. Beslon, F. Biennier, and B. Hirsbrunner, "Multi-robot path-planning based on implicit cooperation in a robotic swarm," in *Proceedings of the second international conference on Autonomous agents*. ACM, 1998, pp. 39–46.

[3] R. Groß, M. Bonani, F. Mondada, and M. Dorigo, "Autonomous self-assembly in swarm-bots," *Robotics, IEEE Transactions on*, vol. 22, no. 6, pp. 1115–1130, 2006.

[4] S. Kumar, R. Chaudhary *et al.*, "Optimization of routing algorithms in ad-hoc networks using swarm intelligence," in *Information and Communication Technologies (WICT), 2011 World Congress on*. IEEE, 2011, pp. 677–681.

[5] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks*, ser. Applied Mathematics Series. Princeton University Press, 2009, to appear. Electronically available at http://coordinationbook.info.

[6] M. Lewis, J. Wang, and P. Scerri, "Teamwork coordination for realistically complex multi robot systems," in *NATO Symposium on Human Factors of Uninhabited Military Vehicles as Force Multipliers*. Citeseer, 2006.

[7] P. Walker, S. Nunnally, M. Lewis, A. Kolling, N. Chakraborty, and K. Sycara, "Neglect benevolence in human control of swarms in the presence of latency," in *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*. IEEE, 2012, pp. 3009–3014.

[8] J. Bachrach, J. Beal, and J. McLurkin, "Composable continuous-space programs for robotic swarms," *Neural computing & applications*, vol. 19, no. 6, pp. 825–847, 2010.

[9] C. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4. ACM, 1987, pp. 25–34.

[10] I. Couzin, J. Krause, R. James, G. Ruxton, and N. Franks, "Collective memory and spatial sorting in animal groups," *Journal of theoretical biology*, vol. 218, no. 1, pp. 1–11, 2002.

[11] I. D. Couzin, J. Krause, N. R. Franks, and S. A. Levin, "Effective leadership and decision-making in animal groups on the move," *Nature*, vol. 433, no. 7025, pp. 513–516, 2005.

[12] S. Amirpour Amraii, N. Chakraborty, and M. Lewis, "Studying direct and indirect human influence on consensus in swarms," in *2012 AAAI Fall Symposium Series*, 2012.

[13] P. Walker, S. Amirpour Amraii, N. Chakraborty, M. Lewis, and K. Sycara, "Human control of leader-based swarms," in *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*. IEEE, 2013.

[14] K. Xu and M. Gerla, "A heterogeneous routing protocol based on a new stable clustering scheme," in *MILCOM 2002. Proceedings*, vol. 2. IEEE, 2002, pp. 838–843.