Human Control of Robot Swarms with Dynamic Leaders*

Phillip Walker[1], Saman Amirpour Amraii[1], Nilanjan Chakraborty[2], Michael Lewis[3], and Katia Sycara[2]

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. This may be necessary in a environmental exploration task where areas of interest arise dynamically, and thus the human operator needs to guide the swarm to explore them. Past work has focused on influencing the swarm via statically selected leaders—swarm members that the operator directly controls. These leaders have been pre-selected and remain leaders throughout the scenario execution. This paper investigates the use of a small subset of the swarm as robot 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 (a) the effect of density of leaders on the ease of human control and system performance, and (b) how restriction of information communicated to the human operator affects the ability to guide the swarm to goal regions. The density of leaders is computed based on an extension of the RCC algorithm used in wireless sensor networks to select cluster heads. We used a "hop guarantee" in this leader selection algorithm as a measure of leader density. A n-hop guarantee means that every robot is at most n-hops away from a leader. In particular, we studied the effect of 1-hop, 2-hop and 3-hop guarantee on the swarm performance. 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 and 3-hop guarantee was not statistically significant. Furthermore, we found that performance was just as good when the information returned to the operator was restricted, showing that operators can still navigate a swarm even when they have imperfect information.

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.

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.

[1]Phillip Walker and Saman Amirpour Amraii are students at the Intelligent Systems Program, University of Pittsburgh, Pittsburgh, PA 15260, USA `pmwalk@gmail.com, samirpour@acm.org`

[2]Nilanjan Chakraborty and Katia Sycara are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA `nilanjan@cs.cmu.edu, katia@cs.cmu.edu`

[3]Michael Lewis is with the School of Information Sciences, University of Pittsburgh, Pittsburgh, PA 15260, USA `ml@sis.pitt.edu`

Because of the need to make individual swarm members as cheap 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 a 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 and (b) information communicated to the operator 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 human-to-swarm communication restricted to a small subset of dynamically-selected swarm leaders. Furthermore, we show that different numbers of leaders can be selected, thus tailoring the selection to the needs of particular applications. Additionally, our results show that restricting information communicated to and from the operator regarding the swarm state does not have significant deleterious effects on performance.

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 one of the propagation methods from [12]–the consensus method—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, and 3-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).

We also had different conditions to study the influence of available information to the operator on the system performance. In the *full information conditions*, the operator could see the full state of the swarm, consisting of each robot's position and orientation in the environment. We studied three conditions of full information, namely full information with 1-hop, 2-hop, and 3-hop guarantees. In the *restricted information* conditions, irrespective of the number of hop guarantees, the operator could only see the parts of the swarm that the leader could directly observe—their own positions and orientations, as well as those of their neighbors. We still allowed the operator to see the entire field of obstacles and a goal region so that their absence would not be a confounding factor in the analysis of swarm control. This reflects a scenario where the human may have archived information about the environment beforehand (such as a building blueprint or satellite image). Note that there is no information restricted condition for 1-hop guarantee, since every robot is either a leader or a neighbor of a leader, thus having full information. Therefore, there are two restricted information conditions, i.e. 2-hop and 3-hop with restricted view. This gives a total of 5 conditions we studied.

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. On the other hand, from [15] we know that humans can sometimes still effectively influence a swarm with restricted information. Thus, we hypothesize that restricting the information returned to the operator in this study will not cause a statistically significant drop in performance between the conditions.

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 III we present and discuss the results of the study, and in 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 leaders via continuous velocity (heading and speed) commands. All the leaders received the same velocity commands from the operator, and they influence the rest of the swarm through biasing the alignment vectors of their neighbors performing a flocking algorithm, as described in Section II-B. The main task for the participants is to use the leaders to guide the swarm to a circular goal region (see Figure 2). The goal region moves to a new random position each time the swarm reaches it, allowing the number of times the goal is reached to act as a natural measure of the operator's ability to control the swarm overall. We will show in Section III that the goal region was placed sufficiently randomly as to ensure no condition had an advantage over another due to random chance.

We investigate what effect changing the guaranteed number of hops from any one robot to a leader has on the ability of the operator to control the swarm. This hop guarantee is an easy way for us to give different leader densities in a scalable manner. In addition, we investigate what effect limiting the information returned to the operator has on performance. In particular, we restrict the information returned to the operator to just what the leaders can directly observe (i.e. the leaders themselves and robots one hop away). The participants completed the full information and the information restricted conditions under different 1, 2, and 3-hop guarantees, as will be described later in Section II-D. There is no information restricted condition for the 1-hop case because each robot is already either a leader or the neighbor of a leader; therefore, there were five conditions in total.
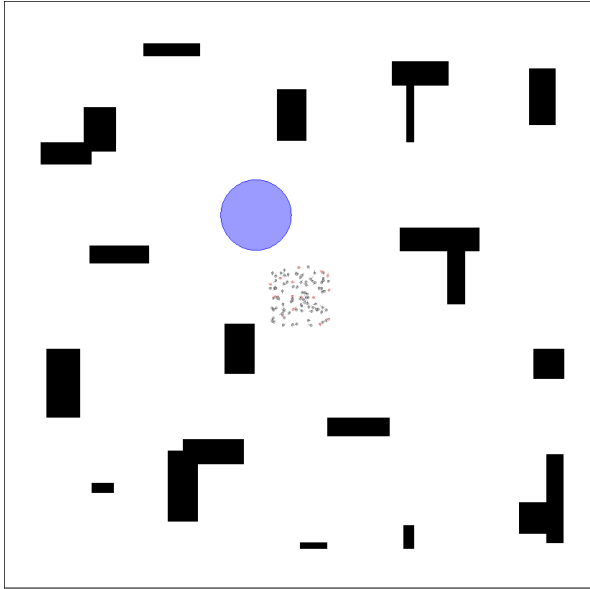

Fig. 1. An example environment with the initial state of the swarm and a sample goal region(shaded circle) shown to scale. Black regions denot obstacles.

A. The Environment and Robots

We use obstacle-filled 100x100 meter environments for the study (see Figure 1), and Stage v.3.2.2 [16] to simulate the 100 robots within these environments. Each environment contains only rectilinear obstacles, at least 10 meters apart and no longer than 20 meters in length, shown as black regions in Figure 1, and each contains similar proportions of obstacle-filled to total space. The goal region is a circle with a diameter of 12 meters (see Figure 1), and was always generated in open space (never overlapped with obstacles). Each participant sees a different, randomly chosen world in each of the five conditions.

The robot controllers and user interface are implemented using the Robot Operating System (ROS) [17]. Each robot has a differential drive, 0.25 x 0.25 meter body, a unique ID number from 0-99, and is equipped with a simulated neighbor sensor to determine the speed and heading of neighboring robots within 4 meters. Finally, every robot also has line-of-sight communication up to 4 meters. This method of communication allows for cases where communication may make use of the visual channel—such as LEDs on-board the robots.

Simulated noise sampled from the Gaussian distribution $\mathcal{N}(0, \frac{\pi}{3})$ is added to the ground truth heading, θ, of each robot when sensed by neighboring robots. This simulates the sensor error that would exist for a swarm operating in the real world with low-cost sensors. In each of the five 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. The maximum speed of the leader robots was $1.0m/s$, and the maximum speed of the non-leaders was $0.75m/s$, a difference that allows the participants to move the leaders around relative to the position of the rest of the swarm. The maximum angular speed of each robot was $2\pi \ rad/s$.

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 2). Once 20% of the swarm (20 robots) are in the goal region simultaneously, the goal will disappear move to a new random position. We used 20% 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. No goal regions are given outside the central 75x75 meters of the environment. This is to ensure participants will not spend the majority of their time waiting for the swarm to traverse the entire width of the environment to reach a goal region. However, the environment is kept at 100x100 meters to ensure proper space to turn and maneuver the swarm, should the participant lose control.

To influence the swarm, the operator is given a virtual joystick with which to steer the leader robots (Figure 2). This joystick allows the participants to set both the heading and speed of all the leaders simultaneously. The other robots
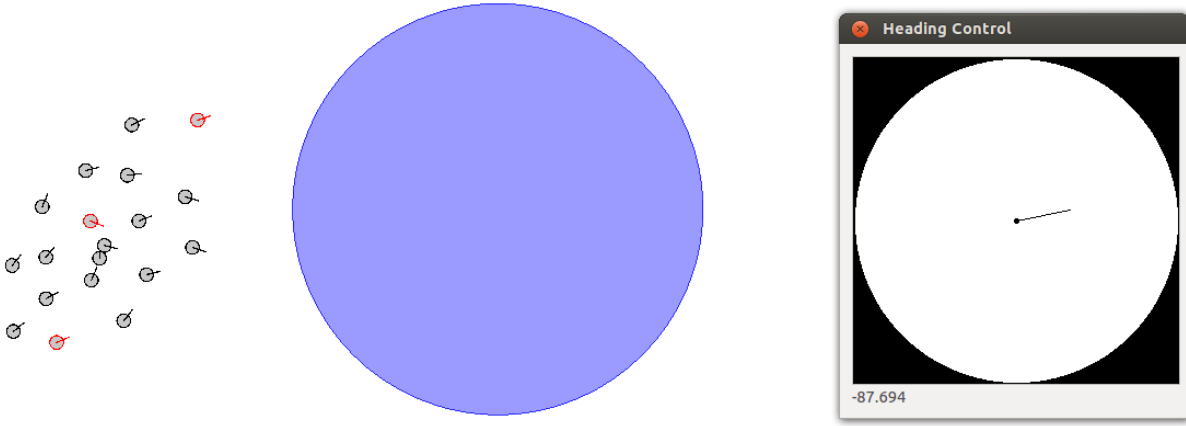
Fig. 2. The swarm of robots (left) is steered to the goal region (center) by the participant teleoperating the leader robots (shown in red) using the virtual joystick (right).

move according to local control laws, hereafter called the *align*, *attract*, and *repel* laws, which are explained below. Only the headings of the vectors output by these laws are used; the velocity of a robot is determined solely by averaging the velocities of its neighbors.

1) Speed and Alignment: To compute the alignment vector, each robot first averages the speed and heading, in the robot's coordinate frame, of each neighbor n_i in the set of neighbors N it can sense. The robot then sets its alignment vector to the average heading of all neighbor robots accordingly:

$$\langle x_a, y_a \rangle = \frac{\sum_{i=1}^{N} \langle \cos{(h_i)}, \sin{(h_i)} \rangle}{|N|}$$

Avove, h_i is the heading of the i-th neighbor, in the robot's coordinate frame. The velocity v is set to the average speed of all neighbor robots. There is no special status given to the neighbors that are leaders. While this may seem to be ignoring useful information, our previous paper [12] suggested that matching a leader's heading and speed directly can cause significant error to be propagated through the swarm. Therefore, we decided to treat leaders identically to other 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 2.5 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 repulsion threshold of 1.5 meters. 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.

$$\langle x_g, y_g \rangle = \alpha \langle x_a, y_a \rangle + \beta \langle x_c, y_c \rangle + \gamma \langle x_r, y_r \rangle \quad (1)$$

Here, α, β, and γ represent constants to give different weights to each of the vectors. For this study, we used

the parameters of $\alpha = 1.5$, $\beta = 2.0$, and $\gamma = 1.0$, which provided some space between the robots without immediately breaking up the swarm.

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 the 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 $t = 2$ seconds across all conditions. Psuedocode for the following algorithm can be found in Algorithm 1.

When the algorithm begins, the robot starts two timers. First, $T_{broadcast}$ (Line 2), which goes off with a period of 0.5 and triggers a send message event (Lines 6-10). The second, T_{leader} (Line 3), represents the current time remaining until the robot switches its leader state (Lines 11-19). When the leader timer does expire, the timer is restarted at t seconds (Lines 12-13), and any robot that is currently a leader will reinitialize its state value $s = x$ (Lines 14-15). If the robot is not a leader, it will become one if it has at least three neighbors in its set of neighbors, N. (Lines 16-18). 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 value 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 value $s_i < x - 1$, then this

robot will defer leadership to the other robot's leader, set its own state value 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 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.

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: Start Timer $T_{broadcast} = 0.5$
3: Start Timer $T_{leader} = t$
4: State $s = n$
5: **while** $true$ **do**
6:   **if** $T_{broadcast} = 0$ **then**
7:     $broadcast(id, s)$
8:     $T_{broadcast} = 0.5$
9:     Start Timer $T_{broadcast}$
10:   **end if**
11:   **if** $T_{leader} = 0$ **then**
12:     $T_{leader} = t$
13:     Start Timer T_{leader}
14:     **if** $s = 0$ **then**
15:       $s = x$
16:     **else if** $|N| > 2$ **then**
17:       $s = 0$
18:     **end if**
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**

D. Conditions and Experiment

Each participant completed five different conditions of the study. Two conditions investigated a 2-hop and 3-hop guarantee with information restriction. This information restriction meant that the participants would only see the information directly observable by the leader robots (i.e., leaders and robots one hop away). We included the three full information conditions, with 1-hop, 2-hop, and 3-hop guarantees, respectively, to see what effect information restriction had on performance. These full information conditions differed from the information restricted conditions only in that they allowed the participant to see all robots, regardless of the current hop guarantee. Again, because the 1-hop condition shows all information already, there is no separate information-restricted condition for the 1-hop guarantee. 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 2). Once 20 or more of the 100 robots reached the goal region, it would immediately move to a new random position free of obstacle overlap.

Each participant was given a short instruction session, in which the experimenter 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 on each condition to train and understand the controls and properties of the experiment unique to that condition. Finally, each participant had 10 minutes to reach as many goal regions as possible for each of the five conditions in a random order. The conditions had different, random sets of obstacles in the environment (see Figure 1 for an example). There were 17 (7 female and 10 male) participants from the University of Pittsburgh and the surrounding region, and each was compensated for their time in the study.

III. RESULTS AND DISCUSSION

During the study, each robot's position, heading, and current state was logged each second for data analysis, as were the positions of the goal regions and when participants reached them. We used the number of goals reached over the course of the ten minutes for each condition as the main performance measure. For the tests given below, M indicates the mean value, F the F-statistic of the ANOVA, t the t-statistic of the t-test, and p the p-value of the performed test. Before proceeding with data analysis, we first verified that there were no significant differences between conditions for the average distance to a new goal. A one-way ANOVA comparing the average distance to a new goal across conditions shows that there were no significant differences ($F = 0.45$, $p = .770$).

In our first test, we performed a one-way ANOVA to analyze the performance differences in operator performance across conditions. The results show a significant difference ($F = 8.72$, $p < .001$, see Figure 3), with participants showing significantly better performance in the 1-hop condition ($M = 8.65$ goals reached) than the full information 2-hop ($M = 5.18$, $t = 4.80$, $p < .001$), full information

3-hop ($M = 5.76$, $t = 4.09$, $p < .001$), information restricted 2-hop ($M = 5.53$, $t = 3.71$, $p < .001$), and information restricted 3-hop ($M = 5.53$, $t = 3.94$, $p < .001$) conditions. However, we found no significant differences between any other two conditions, nor did we find any significant difference found between the 2-hop and 3-hop conditions overall ($t = 0.68$, $p = .499$).
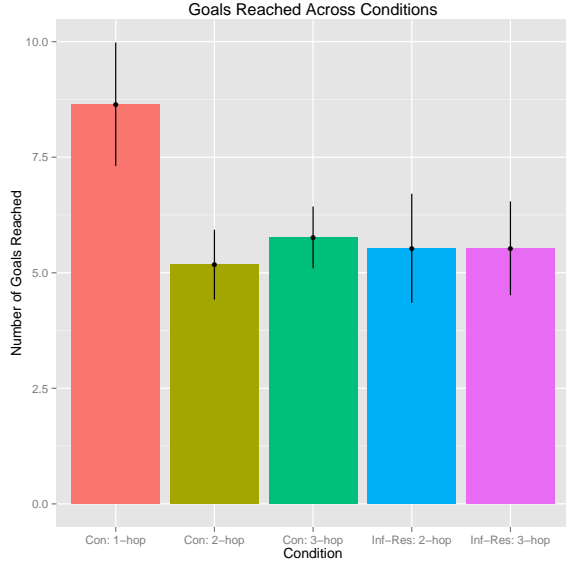

Fig. 3. Then number of goals found on average across each of the five conditions.

Further investigation shows that there were no significant differences between the overall tracking error (that is, the difference between the swarm's average heading and the last heading command given by the participant) in the 2-hop and 3-hop conditions ($t = 1.107$, $p = .273$). This means that the participant's input heading had no less of an influence with a 3-hop guarantee than with a 2-hop guarantee. We see this despite the fact the robots exactly three hops away from a leader had a larger error than those exactly two hops away ($t = 8.487$, $p < .001$). This is explained by the fact that, in the 3-hop conditions, there were only on average 2.13 robots exactly 3 hops away from a leader at any given time—too small a number to have a significant impact on the swarm's average heading. Furthermore, as seen in Figure 4, while there was a significant difference between the number of leaders in the 2-hop ($M = 5.46$) and 3-hop conditions ($M = 4.49$, $t = 8.923$, $p < .001$), this difference was small, with an average of 5.5 leaders in the 2-hop condition, and 4.5 leaders in the 3-hop condition. This further explains the lack of a performance difference seen between the two conditions. Note that 4 also shows that there were significantly more leaders in the 1-hop condition ($M = 11.82$) than either the 2-hop ($t = 38.86$, $p < .001$) or 3-hop ($t = 42.87$, $p < .001$) conditions.

Despite the fact that no difference was found between the 2 and 3-hop conditions, we know from [13] that influencing a swarm of 20 robots—operating under the same control

algorithm presented here—with only one leader is almost impossible. Therefore, our original hypothesis, stating that performance would degrade as the number of guaranteed hops increases, might be better described as a performance plateau. In other words, performance may initially drop once there is no guarantee that each robot is adjacent to a leader, but stay consistent through higher hop guarantees as long as a sufficient percentage of the swarm are leaders. What percentage is sufficient likely depends on the swarm size (as suggested in [11]), should be investigated in future research.
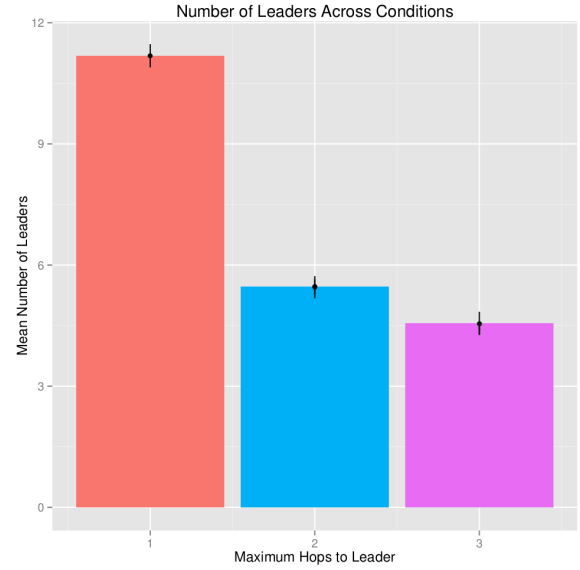

Fig. 4. The average number of leaders between different hop guarantees.

We then investigated performance differences between the full information and information restricted conditions. We found no effect for restricting information ($t = 0.136$, $p = .893$), which was consistent with our hypothesis. This result may initially seem surprising to some readers; however, as found in [15], human operators can still successfully influence a swarm when they are given only the centroid of the swarm and mean heading. Therefore, it is not difficult to see why participants would have similar success when a small subset of the swarm is not visible.

In order to investigate what effect the differences in operator behaviors have on the swarm, we looked at the effect that operator's input speed had on certain swarm characteristics. Here we found contrasting results. As the participants increased their input speed, they reached significantly more goal regions ($r^2 = 0.272$, $p < .001$, see Figure 5). However, increased speed also lead to a decrease in the number of robots in the main connected component of the swarm ($r^2 = 0.190$, $p < .001$, see Figure 6), and an increase in the mean swarm diameter ($r^2 = 0.346$, $p < .001$, see Figure 7). This means that increasing the swarm speed had the contrasting effect of increasing the number of goals reached, while also making the swarm more disconnected and taking up a larger footprint. Thus, moving at a higher speed may not be sustainable during longer operations. Furthermore, a

spatially larger swarm seemingly would be harder to both maneuver around obstacles and to also have 20 robots in the goal region once reached. Another cause of increased swarm diameter was the change in heading given by the participant—results show that as the participant performed larger turns with the swarm, the diameter grew ($p < .001$, $r^2 = 0.007$).
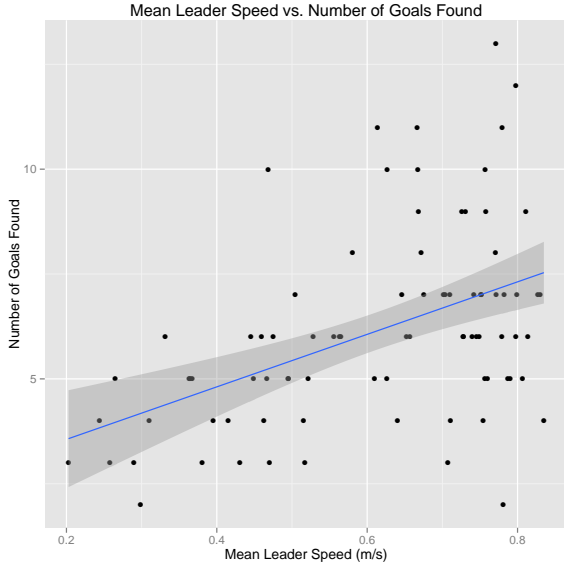

Fig. 5. Comparing the operator's input speed with the number of goals reached.
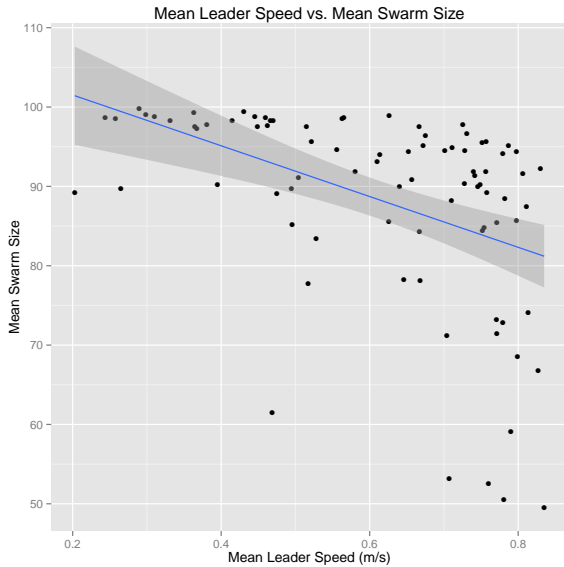

Fig. 6. Comparing the operator's input speed with the number of robots in the main swarm.

We did not find any difference between the conditions in terms of mean input speed ($F = 0.774$, $p = .545$) or the size of turns ($F = 1.618$, $p = .167$), meaning that participants behaved similarly across conditions in this respect.
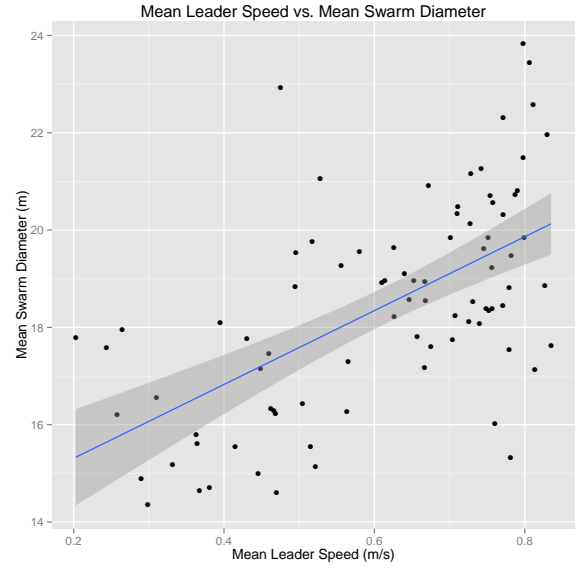

Fig. 7. Comparing the operator's input speed with the diameter of the main swarm.

IV. CONCLUSION

Overall, the results show that influencing a swarm via dynamic leaders is a viable method of swarm control, with performance degrading once one moves away from a guarantee that each robot is at most one hop from a leader. Once this 1-hop guarantee is relaxed, however, there seems to be an initial plateau where performance remains the same, despite slightly different leader densities. In the experiment presented here, this was likely due to the fact that the 2-hop and 3-hop conditions were not that dissimilar. Further work should investigate whether this plateau would hold for larger swarms, and whether the results found herein are a feature of the particular swarm size used. Future research using larger swarms could also expand on this to map a performance curve, from 1 to n hops, given a certain swarm size. An interesting analysis would be to see how these curves differ as the size of the swarm increases.

Another finding of this study is that restricting information about the swarm state seen by the operator to only the local neighborhoods of the leader robots has no effect on the ability of participants to influence the swarm. This shows that aggregating information about each robot throughout the swarm to send back to the operator may be unnecessary. This agrees with the results in [15]. However, further research could also investigate whether information restriction becomes more inhibiting with larger hop guarantees in larger swarms, or with other swarm behaviors besides flocking to a goal.

In summary, we have shown that it is possible to control a simulated swarm of robots in environments with obstacles and dynamically arising goal regions by restricting human-to-swarm communication to a small subset of swarm leaders, and that these leaders can be dynamically selected as the swarm moves. Furthermore, we've shown that different num-

bers of leaders are possible by changing the leader selection algorithm, so that the algorithm could be tailored to the specific application of the swarm.

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.

[15] S. Nunnally, P. Walker, A. Kolling, N. Chakraborty, M. Lewis, K. Sycara, and M. Goodrich, "Human influence of robotic swarms with bandwidth and localization issues," *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, pp. 333–338, 2012.

[16] B. Gerkey, R. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Proceedings of the 11th international conference on advanced robotics*. Portugal, 2003, pp. 317–323.

[17] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, vol. 3, no. 3.2, 2009.