Investigating Neglect Benevolence and Communication Latency During Human-Swarm Interaction[*][†]

Phillip Walker, Steven Nunnally, Michael Lewis
University of Pittsburgh
Pittsburgh, PA

Andreas Kolling, Nilanjan Chakraborty, Katia Sycara
Carnegie Mellon University
Pittsburgh, PA

Abstract

In practical applications of robot swarms with bio-inspired behaviors, a human operator will need to exert control over the swarm to fulfill the mission objectives. In many operational settings, human operators are remotely located and the communication environment is harsh. Hence, there exists some latency in information (or control command) transfer between the human and the swarm. In this paper, we conduct experiments of human-swarm interaction to investigate the effects of communication latency on the performance of a human-swarm system in a swarm foraging task. We develop and investigate the concept of *neglect benevolence*, where a human operator allows the swarm to evolve on its own and stabilize before giving new commands. Our experimental results indicate that operators exploited neglect benevolence in different ways to develop successful strategies in the foraging task. Furthermore, we show experimentally that the use of a predictive display can help mitigate the adverse effects of communication latency.

Introduction

Swarm robotic systems are composed of simple individual units and generate collective behavior that is robust to failure of individual robots (Reynolds 1987; Couzin et al. 2002). However, for practical use of such systems in large and complex human-supervised missions, key problems that arise in human-swarm interaction need to be understood and solved. In application scenarios, the human operator may be remotely located and there may be communication constraints due to the hardware limitations of the robots (e.g., communication radios of limited power) and the environmental properties (e.g. underwater environments). This will lead to delay in the communication of information between the swarm and the human. The delay in communication results in the human neither knowing perfectly the current state of the swarm nor the effect of her action on the robots. The extant literature on Human-Swarm Interaction (HSI) (Bashyal and Venayagamoorthy 2008; Kolling, Nunnally, and Lewis 2012; Coppin and Legras 2012; Cummings 2004; Klarer 1998; Kira and Potter 2009; Naghsh et al. 2008; Goodrich et al. 2011) has not studied the performance and behavior of human operators in the presence of delayed information transmission between the swarm and the human and vice versa. Therefore, in this paper, we create an experimental scenario to study the effects of latency and error on human performance in controlling swarm robotic systems. We also study the use of predictive displays to mitigate the effect of latency.

In our experimental foraging scenario, a human operator guides a swarm to find unknown targets in a given area. The robots have a single behavior, namely flocking, and the operator applies inputs (a) to give a desired direction of flocking to the robots and (b) to enforce cohesiveness among the robots (by activating constraints for attracting neighbors that are far away and repelling neighbors that are very close). In our experiment, each subject performs the mission under three conditions, namely, (a) without any latency (control condition), (b) with equal latency in the human to swarm and swarm to human communication channel (c) the same latency as (b) but with a predictive display. In all conditions, each robot has some error in transforming the orientation heading to its own reference frame (due to localization errors), which is modeled as a Gaussian distribution. Our experimental results indicate that, as expected, there is a degradation of performance due to latency. However, when using the predictive display, the performance of the operators can be as good as it was in the absence of delay (control condition). We also found that the users exhibited different strategies for effectively controlling the swarm.

The human operator needs to influence the swarm without adversely disturbing the swarm (such as breaking it into many small connected components). The effect of an operator command is dependent on swarm state, which gradually evolves to a steady state after a command has been issued. To capture the idea that humans may need to observe the evolution of the swarm state before acting, we investigate a novel concept called *neglect benevolence*, whereby neglecting the swarm to allow for stabilization before issuing new commands may be beneficial to overall mission performance. An analogous but dif-

[*]This research has been sponsored in part by AFOSR FA955008-10356 and ONR Grant N0001409-10680.

[†]An earlier version of this paper has been submitted to the Swarm, Evolutionary and Memetic Computing Conference (SEMCCO 2012)

ferent concept called *neglect tolerance* (Xu et al. 2010; Dai, Sycara, and Lewis 2011) is used in human robot interaction. For independently operating multiple robots, neglect tolerance is defined as the time a human can neglect a robot without degradation in system performance. For neglect tolerance, it is assumed that the performance of an individual robot degrades with time and hence the attention of the operator needs to be scheduled so that the time between servicing robots is minimized (Mitchell and Cummings 2005; Mau and Dolan 2007). In contrast, neglect benevolence captures the concept that it may be beneficial to leave the swarm alone for a certain length of time after issuing an instruction to allow the behavior to stabilize (since the swarm state may not degrade monotonically with time). Our results show evidence of neglect benevolence.

Task Description

Our study investigates the ability of a human operator to effectively influence a swarm operating under algorithms that require time to exhibit emergent behaviors. In particular, we investigate (1) the effect of communication latency in human-swarm performance, (2) the effect of predictive displays, and (3) the existence of neglect benevolence as a new notion in HSI. We designed a foraging task that requires users to direct a swarm around an open environment using instructions to change swarm heading and flocking constraints. We also use this study to look at the effect of communication latency on this ability.

The Environment

The overall task of the experiment is to guide the swarm around an open, 100x100 meter environment to find targets of different colors. We use three different environments divided into six regions, with each region containing one of three target frequencies: *low* (0-4 targets), *medium* (5-9 targets), or *high* (10+). The target distribution is different across the search missions that each participant solves, but each environment contains 1 high, 2 low, and 3 medium frequency regions. There are 40 targets in total in each of the three environments, and each participant receives a worksheet indicating the target frequency of each region.

We use Stage v. 3.2.2 (Gerkey, Vaughan, and Howard 2003) to simulate the environment, the targets, and a swarm of 40 differential drive P2AT robots. Robot controllers are implemented using the Robot Operating System (ROS) (Quigley et al. 2009). Each robot is equipped with a color sensor with a range of 4 meters to detect the colored targets. We also simulate an additional sensor that allows the robots to sense the location of a neighbor within 4 meters. This allows each robot to estimate the direction of motion of its neighbors from repeated observations of their location.

The graphical user interface is also implemented in ROS. This interface displays the known targets and positions of the robots; however, it does not display the region boundaries. During the trial, each robot transmits its position and observations from its color sensor to the user interface. A target is considered found only if six or more robots detect it simultaneously, at which point the target is shown on the map and a counter on the side is incremented.

Human Influence

Users can influence the swarm with three commands: *stop*, *heading*, and *apply-constraints*. The *stop* command simply instructs the robots to stop their motors. The *heading* command broadcasts a global heading to the swarm. To simulate a localization error, every robot interprets the global heading with respect to a local coordinate frame to compute its goal heading. The orientation of this local coordinate frame differs from the true orientation of the robot by an error sampled from the Gaussian distribution $\mathcal{N}(0, \frac{4\pi}{9})$.

Upon receiving the command, the robots turn toward their respective goal heading and begin moving (Figure 1a). In order to correct for the erroneous interpretations of the global heading, each robot also executes a consensus algorithm at a frequency of 0.5 Hertz. Robots sense the direction of motion of their neighbors and update their goal heading to the average goal heading of their neighbors and themselves.

By using the consensus algorithm, robots will change their heading to the average heading of their neighbors, and all robots in a connected component of the swarm will eventually move in the same direction (Figure 1b). The amount of time needed to reach consensus depends on the spectral properties of the connectivity graph of the robots (Xiao, Boyd, and Lall 2005). If each robot is connected to every other robot, then the consensus happens in one cycle. However, in general, the robot connectivity graph is not complete and may not even be connected as the robots move. In such cases, there will usually be a bias in the heading of the swarms when the headings converge.

Finally, the user can issue an *apply-constraints* command, which applies biologically-inspired flocking constraints similar to those in (Reynolds 1987; Couzin et al. 2002; Goodrich et al. 2011). These constraints force robots to repel from each other if they are closer than 1.5 meters, and otherwise attract to neighbors further than 3 meters. Only the closest 5 neighbors are considered for these constraints. If a robot is between 1.5 and 3 meters from each of it's neighbors, it proceeds toward the goal heading dictated by the consensus algorithm (Figure 1c).

Applying the constraints serves two necessary functions: the repulsive force spaces out the swarm to give better coverage, and the attractive force helps prevent the swarm from splitting into many disconnected subgroups. However, these constraints were not automatically on for the duration of the study because of the need for a swarm to have time to reach consensus. The consensus algorithm required robots to sense the positions of their neighbors over time in order to get accurate heading estimates. Therefore, if constraints were applied automatically following a heading command, much of this movement would be due to enforcing the 1.5 to 3m separation between robots, which would introduce a significant amount of noise to the consensus algorithm. Allowing the operator to activate constraints provides the opportunity to observe the swarm and decide when constraints become necessary.

Experimental Design

The experiment consists of three conditions—the *control*, *latency*, and *predictive* conditions. In all conditions, the op-

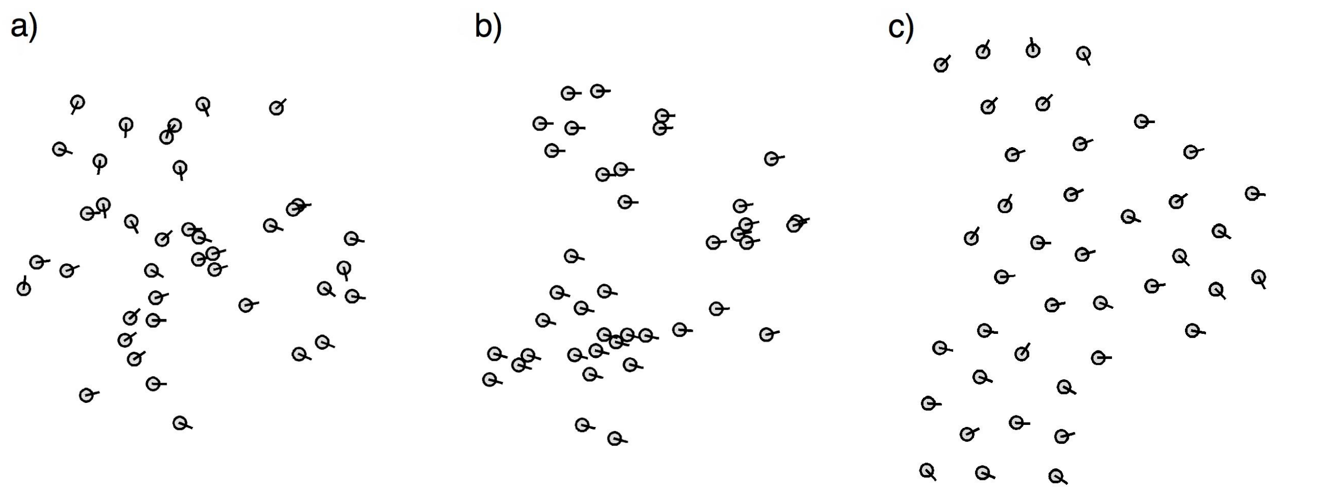


Figure 1: The swarm in each of the three possible states: when the user first issues a heading command (a), after the consensus algorithm has stabilized (b), and after the user has issued the constraints command and the robots have adjusted (c).

erator begins with a swarm of 40 robots positioned randomly in a 10x10 meter box.

In the *control* condition, there is no latency in either of the human-to-swarm or swarm-to-human channels.

In the *latency* condition each channel—operator-to-swarm and swarm-to-operator—has a latency of 10 seconds, meaning that operator-issued commands will not reach the robots until 10 seconds after issuing, and the state of the swarm displayed in the interface for the user is 10 seconds old. Therefore, from the operator's point of view, the swarm will not begin executing the command until 20 seconds after the *heading* instruction is issued, as the message will take 10 seconds to reach the swarm, and the reflection of this command will take 10 more seconds to return to the operator.

In the *predictive* condition, the latency remains present; however, the interface gives the user a prediction of where each member of the swarm will be in 20 seconds (the time it takes for an operator's command to travel to the swarm and the result to return to the operator) by taking the heading and speed (which is a constant 0.5 m/s) of each swarm member and extrapolating the robot's position that far in the future (Figure 2). The prediction does not extend past the point where the robots would perform the user's command. In other words, if the user issued a command 3s ago, the prediction will only show the swarm state 17s further into the future (which is 7 seconds ahead of the actual state).

Each participant has a different environment for each of these conditions, and the order of both the conditions and the maps are randomized for each participant in order to remove any learning biases. 21 participants (8 men and 13 women) were recruited from the greater Pittsburgh area to participate in the study. Each participant was given a short explanation of the controls and goals of the study and a 10-minute training session to familiarize themselves with the interface, after which they completed each of the three conditions.

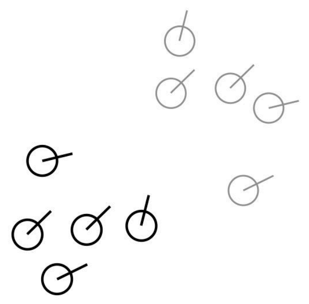


Figure 2: An illustration of the predictive display condition. The interface projected a lighter shadow ahead of each robot to predict for the user where the robot would be in 10 seconds, or when the next command is received.

Results and Discussion

The overall mission performance for each participant is measured in terms of the number of targets found and the coverage of the high-frequency target regions. In the *control* condition participants found 19.86 targets and covered $1.47m^2/s$ of the high-frequency target regions on average, both of which were significantly higher than in the *latency* condition, where participants found 16.71 targets ($p = .021$) and covered $0.98m^2/s$ of the high-frequency regions ($p = .007$). In the *predictive* condition, however, participants found 18.86 targets on average and covered $1.24m^2/s$ of the high-frequency regions, neither of which were significantly different from the *control* condition ($p = .467$ and $p = .196$, respectively). These results show that the latency of 10 seconds significantly impedes performance, but that the predictive display in the *predictive* condition removes this impediment.

An indirect measure of an operator's ability to control the swarm is the swarm's overall connectivity. To determine the overall connectivity of the swarm at any given time, we represented the swarm as a simple graph, G, and used the second eigenvalue of the graph's Laplacian as the connectivity measure. Keeping the swarm connected has two benefits. First, such a swarm is less likely to break off into smaller connected components, which allows the user to meet the six-robot threshold for sensing a target more easily. Secondly, a highly-connected swarm will reach consensus faster, as each robot will have more neighbors to average with during each consensus round.

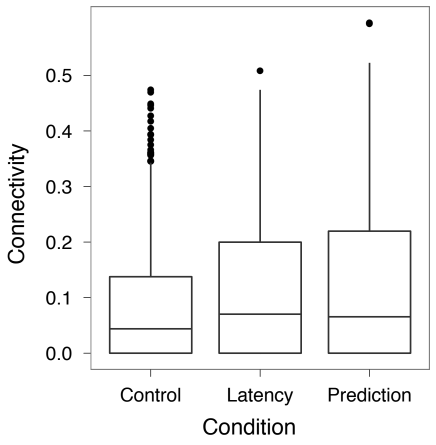
We see that the *latency* condition had an average connectivity of 0.111, which was significantly higher than in *control* condition, which had an average connectivity of 0.084, $p < .001$. Similarly, average connectivity in the *predictive* condition was also significantly higher than in the *control* condition, with a value of 0.116, $p < .001$, see Fig. 3. This points to the existence of neglect benevolence, as it demonstrates that communication latency helped enforce swarm connectivity by causing operators to wait to see the results of their heading command before deciding whether to issue a new one. As a consequence, each command has a longer duration, thus giving the swarm more time to stabilize after each command. We find this is indeed true, with users issuing significantly more commands on average in the *control* condition ($M = 27.81$) than in the *latency* condition ($M = 17.76$, $p = .028$), and significantly more than in the *predictive* condition ($M = 18.86$, $p = .052$).

To investigate the various behaviors and strategies of the operators, we investigated the duration and timing of the *heading* and *apply-constraints* instructions. We analyzed the average time between a *heading* and a subsequent *apply-constraints* command (hereafter referred to as time to constraints), or the next *heading* command (duration). Because these instructions involve altering the current state of the swarm at the time they are issued, even two identical commands (i.e., two heading commands with the same heading) can lead to drastically different effects depending on the state of the swarm.

As demonstrated by the previous results, because the number of heading commands were different across condition, the duration of the commands are similarly different. The *apply-constraints* instruction, however, can be more flexible, and operators may decide to issue constraints at any point in time after a *heading* command, or issue a new *heading* command without activating constraints at all. To investigate the effect of the application of constraints, we clustered the data into three equal-sized groups across all missions into a high (100% to 78%), medium (78% to 45%), or low (45% to 0%) group corresponding to the number of heading commands for which constraints were later applied.

Performance in terms of targets found does not differ between these groups, but the total area swept is significantly different, with fewer constraints (low) leading to less overall area covered by at least six sensors ($p = .040$). On the other hand, many constraints (high) lead to a larger error between the heading of the swarm and the operator's goal heading ($p = .011$), and fewer constraints (low) have more heading instructions leading to a consensus (74%, $p < .001$).

These results suggest that operators employ different strategies to find a larger number of targets, with some operators using constraints earlier and more often, increasing coverage at the expense of higher heading errors, while others preferred the opposite.



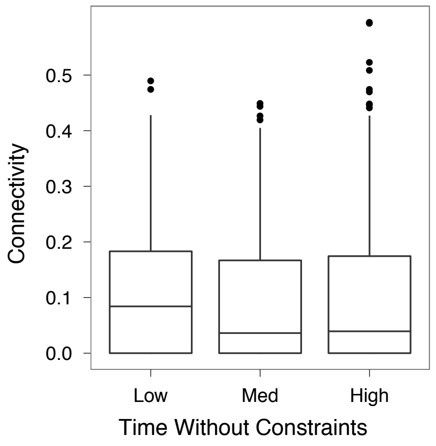


Figure 3: These graphs display the connectivity of the swarm at the end of each heading command issued. Connectivity is significantly different across conditions (left), with the latency conditions generally having more connected swarms. The average time to constraints also impacts connectivity (right), with participants choosing to enforce constraints more often generally having better connected swarms. The boxes in each figure represent the bottom three quartiles, and the outliers are marked as black dots.

We expected a difference in heading duration and time to constraints between the two latency conditions and the control condition, as participants must wait 20 seconds to see the results of their commands. Interestingly, however, across all instructions, only 27% in the *latency* condition and 30% in the *predictive* condition have constraints activated later than 20 seconds, neither of which were significantly different from control, meaning that, unlike with the heading commands, operators often issued the constraints prior to seeing the effect of the heading instruction on the swarm. Swarm connectivity was also significantly impacted by the application of constraints as well, with the high constraints group having significantly better connectivity than the medium or low group ($p = .043$), see Fig. 3.

These results provide considerable support for neglect benevolence. Frequent and short commands provide an operator more control, but sacrifice swarm cohesion as reflected in the lower connectivity value and the higher number of connected components. This is largely due to the fact that new heading commands reintroduce error into the swarm members' estimated heading and require several rounds of consensus to stabilize. Activating the constraints too early and often, however, leads to higher heading errors, and thus may make the swarm more difficult for the human to control. We found that operators develop two different strategies around neglect benevolence: either stabilize the consensus and lower the heading error, or maintain swarm cohesion and improve coverage. It appears operators were able to use either method to their advantage and obtain a good performance, and that, while latency can degrade performance overall, it does not impact one strategy more than the other.

Conclusions and Future Work

This study provides support for the idea of neglect benevolence, with the commands in the study leading to strategies with different costs and benefits depending on the state of the swarm at the time the commands were issued. Frequent heading commands provided the user more control over the direction and location of the swarm at the expense of total coverage and swarm connectivity. Due to the nature of the swarm algorithms, high position and heading accuracy and high swarm cohesion were not possible simultaneously. Therefore, participants had to decide which characteristics were more important. For the present study, both strategies achieved success; however, other tasks may be better achieved with one or the other. This will be the subject of future study.

Latency had a negative effect on the number of targets found; however, using a predictive display mitigated the negative effects. Latency also seemed to significantly impact the frequency with which an operator issues *heading* commands, but not *apply-constraints* commands. As this is the first study to investigate latency in human-swarm interaction, future work will address latency issues for human control of other tasks and swarm algorithms.

References

Bashyal, S., and Venayagamoorthy, G. 2008. Human swarm interaction for radiation source search and localization. In *Swarm Intelligence Symposium, 2008. SIS 2008. IEEE*, 1–8. IEEE.

Coppin, G., and Legras, F. 2012. Autonomy spectrum and performance perception issues in swarm supervisory control. *Proceedings of the IEEE* (99):590–603.

Couzin, I.; Krause, J.; James, R.; Ruxton, G.; and Franks, N. 2002. Collective memory and spatial sorting in animal groups. *Journal of theoretical biology* 218(1):1–11.

Cummings, M. 2004. Human supervisory control of swarming networks. In *2nd Annual Swarming: Autonomous Intelligent Networked Systems Conference*.

Dai, T.; Sycara, K.; and Lewis, M. 2011. A game theoretic queueing approach to self-assessment in human-robot interaction systems. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 58–63. IEEE.

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

Goodrich, M.; Pendleton, B.; Sujit, P.; and Pinto, J. 2011. Toward human interaction with bio-inspired robot teams. In *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*, 2859–2864. IEEE.

Kira, Z., and Potter, M. 2009. Exerting human control over decentralized robot swarms. In *Autonomous Robots and Agents, 2009. ICARA 2009. 4th International Conference on*, 566–571. IEEE.

Klarer, P. 1998. Flocking small smart machines: An experiment in cooperative, multi-machine control. Technical report, Sandia National Labs., Albuquerque, NM (United States).

Kolling, A.; Nunnally, S.; and Lewis, M. 2012. Towards human control of robot swarms. In *Proceedings of the 7th international conference on Human-robot interaction*, 89–96. ACM.

Mau, S., and Dolan, J. 2007. Scheduling for humans in multirobot supervisory control. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, 1637–1643. IEEE.

Mitchell, P., and Cummings, M. 2005. Management of multiple dynamic human supervisory control tasks. In *10th International Command and Control Research and Technology Symposium*.

Naghsh, A.; Gancet, J.; Tanoto, A.; and Roast, C. 2008. Analysis and design of human-robot swarm interaction in firefighting. In *Robot and Human Interactive Communication, 2008. RO-MAN 2008. The 17th IEEE International Symposium on*, 255–260. IEEE.

Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; and Ng, A. 2009. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, volume 3.

Reynolds, C. 1987. Flocks, herds and schools: A distributed behavioral model. In *ACM SIGGRAPH Computer Graphics*, volume 21, 25–34. ACM.

Xiao, L.; Boyd, S.; and Lall, S. 2005. A scheme for robust distributed sensor fusion based on average consensus. In *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, 63–70. Ieee.

Xu, Y.; Dai, T.; Sycara, K.; and Lewis, M. 2010. Service level differentiation in multi-robots control. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, 2224–2230. IEEE.