2016-06-01

A Distributed Control Algorithm for Small Swarms in Cordon and Patrol

C Kristopher Alder
Brigham Young University - Provo, alderkr@gmail.com

A Distributed Control Algorithm for Small Swarms in Cordon and Patrol

C. Kristopher Alder

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Michael A. Goodrich, Chair
Mark B. Colton
Daniel Zappala

Department of Computer Science

Brigham Young University

June 2016

ABSTRACT

A Distributed Control Algorithm for Small Swarms in Cordon and Patrol

C. Kristopher Alder
Department of Computer Science, BYU
Master of Science

Distributed teams of air and ground robots have the potential to be very useful in a variety of application domains, and much work is being done to design distributed algorithms that produce useful behaviors. This thesis presents a set of distributed algorithms that operate under minimal human input for patrol and cordon tasks. The algorithms allow the team to surround and travel between objects of interest. Empirical analyses indicate that the surrounding behaviors are robust to variations on the shape of the object of interest, communication loss, and robot failures.

Keywords: swarm, robots, distributed, cordon and search, robustness, graph theoretic

ACKNOWLEDGMENTS

This work was funded by the United States Army Research Laboratory through contract W911NF-14-1-0633. All statements, findings, and results are the responsibility of the authors and may not reflect the opinions of the funding organization.

I would like to thank my advisor, Dr. Goodrich, as well as my committee members for their suggestions and guidance.

I would also like to thank my fiancée Annie for her support and encouragement.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Teams consisting of both humans and robots can be more effective than either robots or humans working independently. For example, in a cordon-and-patrol scenario, humans search a building for military, police, or other purposes. A team of robots can help the humans performing the search by surrounding the building (forming a cordon) and detecting anyone leaving the area. This allows humans to focus on more attention on searching the interior of the building instead of preventing escape. After the building is successfully searched, human operators may need to re-assign the robot team to travel toward a new location (carry out a patrol), possibly to surround a new building.

Coordinating actions between humans and robot teams is difficult. An effective solution to this problem should allow a human operator to control the robot team with little cognitive overhead. This is important since all effort spent managing the robots distracts from other tasks and that might need to be performed by the operator. Cordon-and-patrol tasks may often be carried out in dangerous or uncertain environments, making this especially important. A distributed algorithm is an important part of solving this problem since it allows human users to influence all of the robots by interacting directly with only a small portion of the team.

Distributed control algorithms for managing robot teams have desirable properties. They allow the team to be robust to hardware failures of individual robots and inconsistent communication due to poor reception or interference. An effective distributed solution also minimizes the need for global information by relying primarily on local sensing and reducing

the amount of communication required between robots. Depending on local sensing also allows the robot team to perform successfully without needing a prioiri knowledge of the world.

The distributed algorithms we created are presented in two papers:

1. C Kristopher Alder, Samuel J McDonald, Mark B Colton, and Michael A Goodrich. Toward haptic-based management of small swarms in cordon and patrol. In *Swarm/Human Blended Intelligence Workshop (SHBI), 2015* pages 1–8. IEEE, 2015.
2. C Kristopher Alder, Michael A Goodrich, and Mark B Colton. Robust behavior design for a team of distributed robots in patrol and search. In *International Conference on Intelligent Robots and Systems (IROS), 2016*. IEEE, 2016. — Under Review

Chapters 2 and 3 are replicated from these papers.

Following the two papers is a chapter containing additional results validating the algorithm. This chapter examines the unique diffusion factor the algorithm uses and shows that it is essential to the system's ability to successfully form a cordon.

Finally, the thesis concludes with a summary and a discussion of future work.

Chapter 2

Toward Haptic-based Management of Small Swarms in Cordon and Patrol

C Kristopher Alder, Samuel J McDonald, Mark B Colton, and Michael A Goodrich. Toward haptic-based management of small swarms in cordon and patrol. In *Swarm/Human Blended Intelligence Workshop (SHBI), 2015* pages 1–8. IEEE, 2015

2.1 Introduction

There are many current and future scenarios in which a human must manage a team of air, ground, and humanoid robots (generically referred to as *agents* in this paper). These scenarios include wilderness search-and-rescue [17], rescue operations in buildings damaged by fire or earthquake [2, 24], searching of a building by law enforcement agencies [21], pollution monitoring and clean-up [22], and military patrol and cordon operations in an urban environment [26]. In each scenario, the agent team serves as an extension of the humans ability to gather information in complex and often dangerous environments, and the tasks are often time-sensitive. Enabling the human operator to manage the agent team in an intuitive, effective, and time-efficient manner is therefore critical to the success of operations involving agent teams.

The state of the art in controlling autonomous agents (in use by current military, law enforcement, and search-and-rescue agencies) is for a single agent to be controlled and monitored by one or many human operators (see, for example, [28]). This interaction model is clearly not ideal if the objective of employing autonomous agents is to augment the capabilities of humans and maximize the information-gathering capabilities of the team [19, 40].

A preferable interaction model is for a single human operator to control multiple autonomous agents. The effectiveness of such an approach is limited by the human's ability to command the actions of multiple agents and receive information about the state of the agent team. When the agent team possesses appropriate autonomy for the given scenario, the problem becomes one of team management rather than agent control, enabling the human to focus on task objectives and interpretation of gathered data, rather than on the agents.

In this paper we describe initial work on an approach to managing a team of agents by a single human user. Our focus is on the patrol and cordon scenario, although the approach will be generalizable to other scenarios where humans must control the movement of teams of autonomous agents, as well as the distribution of the agents within the team and relative to features in the environment. In a patrol and cordon scenario, an agent team works in an urban environment to search buildings. In *surround* mode the user selects a building or buildings for the team to surround, and the team autonomously forms the cordon. The user can see and feel the team distribution, and modify the distribution based on human knowledge of likely entrance and exit points, traffic patterns, obstructions, etc. The user has the ability to modify the size and shape of the cordon. When the user selects a new building or buildings to surround, the team disengages from the original building, autonomously switches into a *travel* mode, and re-forms around the new target.

The approach includes two elements: (1) swarming behaviors that enable autonomous agent teams to self-form around environmental features (buildings) and into travelling formations, and (2) graphical and haptic interactions that enable the user to see, feel, and command the teams location and distribution. The swarming behavior is based on a distributed algorithm that governs the position of and communication between a team of 10–15 agents such that there is minimal need for globally shared information. Graph theory is used to describe relationships between the agents, with each agent corresponding to a node in the graph. During *surround* mode, the agent graphs connections form a spanning ring, whereas the graph contains a spanning forest during *travel* mode, with each agent being

influenced only by a single leading agent. While in *surround* mode the user can feel haptic feedback as he stretches, compresses, or reshapes the spanning ring. In *travel* mode haptic feedback is determined solely by the shape of the spanning forest during movement, so that the user may gain any intuitive sense for the location and overall shape of the team.

The objective of the present work is to create individual and team autonomy behaviors, as well as user interface methods, to enable a single user to manage a planar, swarm-like team in an urban patrol and cordon scenario. Although the work we present is novel, it is not without precedent. Decentralized unmanned ground vehicle (UGV) swarms have been controlled successfully using haptic feedback from a bird's eye view, to maximize manipulability or coverage [31, 33, 37].

In each of these studies, the primary objective was to improve operator performance by decreasing collisions without increasing operator workload.

Results from user studies showed a decrease in task completion time and an increase in team coverage when using haptic feedback. Our work complements this prior work by identifying agent control algorithms that yield, by design, global behaviors that are amenable to haptic interactions.

2.2 Individual Behaviors

In this section, we present the model by which each individual agent in the collective governs its motion. We restrict attention to agents that use as little centralized information as possible. Furthermore, we restrict attention to agents that can sense only two types of objects in the world: other agents and obstacles/points of interest. Based on sensor readings from these types of objects, each agent computes a force that pushes or pulls it in a direction that (a) keeps it close (but not too close) to its neighbors and (b) either keeps it in proximity to the boundary of an object if the agent is in a *surround* mode or helps it avoid an object if the agent is in *travel* mode.

2.2.1 Neighbour Selection

In the literature on bio-inspired collectives, there are two main methods by which an agent determines who influences it [1]: a metric-based method and a k-nearest-neighbors method. The metric-based method is typified by both Couzin's model [10] and Reynolds' model [35] in which agents are influenced by all neighbors within a radius of attraction, repulsion or orientation. The key idea in metric-based methods is that all agents within a certain distance influence the agent, regardless of how many agents are within those neighborhoods.

By contrast, the k-nearest-neighbors (NN) method assumes that each agent can only track a certain number of neighbors, and assumes that the neighbors that are nearest the agent are the ones most likely to exert influence. The NN method is typified by Ballerini's work in which evidence suggested that this model was both biologically plausible and better able to explain the behaviors of starlings than a metric-based method [5].

We adopt the NN method because prior work suggests that it produces collectives that are less likely to fragment in the presence of large perturbations [18]. This means that the forces that an agent experiences are determined by a limited number of agents. This induces what we call an *interagent influence topology*. Later in this paper, we will consider a different topology, which we call the *communication topology* that may allow an agent to send and receive messages from other agents even if those other agents do not directly influence the forces used by the agent to select an action.

Let k denote the number of other agents that an agent is influenced by. Both empirical evaluations and previous work [7] indicate the following trends: larger values of k will cause the swarm to settle on a solution more quickly, while lower values produce swarms that are more flexible and responsive to control by human operators.

It is straightforward to create an algorithm in which each agent identifies its k-nearest neighbors. Let $\mathbf{x}_i$ denote the position of the i^{th} agent in the collective and let $N(\mathbf{x}_i; k)$ denote the positions of the k-nearest neighbors of agent i. From $\mathbf{x}_i$ and $N(\mathbf{x}_i; k)$ distances between an agent and each of its neighbors are easily computed. Let $\mathbf{X}_i$ denote the matrix constructed

from $N(\mathbf{x}_i; k)$ by sorting the neighbor set from nearest to farthest from the agent. Thus,

$$\mathbf{X}_i = [\mathbf{x}_1 \mathbf{x}_2 \dots \mathbf{x}_k] \tag{2.1}$$

where we assume that each $\mathbf{x}_i$ is a column vector and $\mathbf{x}_1$ is the neighbor nearest to agent i, $\mathbf{x}_2$ is the next closest, and so on.

2.2.2 Obstacle Sensing

Recall that we assumed each agent could sense both other agents as well as the positions of obstacles or points of interest in the world. We assume a radar or lidar-type sensor that is able to measure both distance to and direction of these objects. This sensor is consistent with our goal to minimize centralized information in the world because the agents need not understand the location and shape of, for example, a building in the world as long as they can distinguish between a distance measure from a neighboring agent and a distance from an obstacle. As we shall demonstrate, this design frees the user from the need to communicate the location and shape of buildings to each of the agents. It also allows the agents to respond to changing conditions without directly involving the human operator.

In our simulations, each agent sends out radar "pings" at regular intervals in a 360-degree arc. Larger objects, such as walls, will be detected by multiple of these simulated radar "pings". This yields a set of readings of obstacle locations uniformly distributed around agent i, $\{\boldsymbol{\xi}_i^j : j \in \{1, 2, \dots, \frac{360}{\Delta} - 1\}\}$ where Δ is the angle between the different pings. Note that we are using a convention where the symbol for agent-related parameters uses a letter from the English alphabet while the symbol for the corresponding object-related parameter uses the corresponding letter from the Greek alphabet.

We sort these readings from nearest to farthest and to drop readings that are beyond the range of the sensor. This sorted vector of readings is denoted by the array $\boldsymbol{\Xi}_i$. Because objects outside of the sensors' range may not be detected at all, the size of the array may

vary as agents move in the world and objects enter and leave sensor range. Thus,

$$\Xi_i = \left[\boldsymbol{\xi}_i^1 \boldsymbol{\xi}_i^2 \dots \boldsymbol{\xi}_i^n\right] \tag{2.2}$$

where $n < \frac{360}{\Delta}$ is the number of sensor readings within sensor range and $\boldsymbol{\xi}_i^1$ is the location of the sensor ping closest to agent i, $\boldsymbol{\xi}_i^2$ the next closest reading, and so on.

2.2.3 Emphasizing Unique Influences

The first important factor in determining how objects and other agents influence an agent, and an innovative part of the agent controller, is that neighbors or sensor readings that are "unique" should have stronger influence. More specifically, our model limits the amount of influence a small cluster of agents or sensor pings can exert, which tends to keep the interagent topology connected, making it robust as the collective moves through the world or encounters a building.

For each agent $a \in N(\mathbf{x}_i, k)$, we assign a weighting w_i^a which is a function of how unique agent a is relative to other agents. For the nearest agent, which is identified by $a = 1$ since we sorted agents from nearest to farthest, let $w_i^1 = 1$. This initializes the weight-assignment algorithm. For each other agent $a \in \{2, \dots, k\}$, let w_i^a be given by

$$w_i^a = \frac{1}{\pi} \min_{1 \le b < a} \cos^{-1}\left(\frac{\mathbf{x}_a^T \mathbf{x}_b}{\|\mathbf{x}_a\| \|\mathbf{x}_b\|}\right) \tag{2.3}$$

This equation iteratively assigns a weight to every agent $a \in \{2, \dots, k\}$ proportional to the smallest angle between agent a and every other agent that has previously been assigned a weight. The idea is that the normalized inner product given by the fraction gives the cosine of the angle between unassigned agent a and every previously assigned agent b. The arc cosine gives the angle between agent a and agent b, and the smallest such angle indicates how directionally similar that agent is to every previously assigned agent. If there are other, closer agents in the general direction of agent a then agent a is in a cluster of agents and

shouldn't receive a high weighting. The $\frac{1}{\pi}$ normalizes each weight so $w_i^a \in [0, 1]$. The weights are then normalized so that they sum to 1.

Stated simply, agents that lie in a very similar direction to other neighbouring agents are given lower influence.

The same algorithm can be applied to each column in $\boldsymbol{\xi}_i$ yielding a set of "uniqueness" weights for each ping, ω_i^j where $j \in \{1, 2, \ldots, n\}$.

2.2.4 Calculating Force Strength

The weights given by the algorithm described in the previous section determine *how much* an agent in agent i's neighborhood and how much obstacles will influence agent i. We now need to determine *how* agent i's behavior should be influenced by those agents. The introduction identified the second important factor in determining the strength of interaction, namely that, depending on mode, agents want to be close but not too close to other agents and to objects in the world.

Let d_i and δ_i denote the ideal standoff distance that agent i wants to maintain between agents and obstacles, respectively. Given these ideal standoff distances, we can determine how influences from other agents and obstacles push or pull an agent to a new location.

If a neighbor of agent i is too far away then agent i should be attracted to that neighbor. Formally, agent i is attracted to neighbor j when $\|\mathbf{x}_i - \mathbf{x}_j\| > d_i$. We let the strength of the attractive influence grow as the distance increases. Similarly, agents that are closer than the stand-off distance d_i exert a repulsive force that approaches infinity as the distance between the two agents approaches zero. This prevents two agents from colliding. The strength of the force from agent j to agent i is given by

$$s_i^j = w_i^j \left| \frac{1}{d_i} - \frac{1}{\|\mathbf{x}_i - \mathbf{x}_j\|} \right| \tag{2.4}$$

Future work will consider refinements of this equation, but the principles behind it are

simple and are consistent with both Couzin's model [10] and potential-field methods for agent control [4]. The idea is that the repulsive force increases to infinity as the objects approach (as described in the text above) and attractive forces increase as distance increases.

The radar "pings" represented by Ξ_i exert similar repulsive and attractive forces on agent i with strengths scaled by ω_i. This yields the strength of the force from sensor reading j on agent i as

$$\sigma_i^j = \omega_i^j \Big| \frac{1}{\delta_i} - \frac{1}{\|\mathbf{x}_i - \boldsymbol{\xi}_j\|} \Big| \tag{2.5}$$

2.2.5 Agent Control

Agents have two different modes or states: *surround* and *travel*. As described in the next section, agents use a simple form of distributed decision-making to determine when to switch behaviors, and future work should explore other methods (see [36, 39]).

The direction that the agent should travel is given by the sum of the forces derived from other agents and from objects in the world. The previous sections have provided definitions that apply to the most general situation, but some forces are ignored in some of the modes. The most general situation is when the agents are in *surround* mode, in which case the forces on the agent are given by:

$$\begin{aligned} \Delta\mathbf{x}_i &= \sum_{j=1}^{k} s_i^j \frac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|} \Big(\text{sign}(d_i - \|\mathbf{x}_j - \mathbf{x}_i\|) \Big) \\ &+ \sum_{j=1}^{n} \sigma_i^j \frac{\boldsymbol{\xi}_j - \mathbf{x}_i}{\|\boldsymbol{\xi}_j - \mathbf{x}_i\|} \Big(\text{sign}(\delta_i - \|\boldsymbol{\xi}_j - \mathbf{x}_i\|) \Big). \end{aligned}$$

This equation takes a summation over all neighbors (the first line) and all objects (the second line) of a set of unit vectors weighted by the strengths of the objects/neighbors. The unit vectors point in the direction of neighbors/objects, and the sign of these vectors is determined by whether the objects/neighbors attract or repel the agent.

In *travel* mode, two changes are made to this equation: First, only one other agent is considered, a special agent that is leading the group. Second, the agent is never attracted to

an object, only repelled. This yields

$$\begin{aligned}\Delta\mathbf{x}_i &= s_i^\ell \frac{\mathbf{x}_\ell - \mathbf{x}_i}{\|\mathbf{x}_\ell - \mathbf{x}_i\|}\Big(\operatorname{sign}(d_i - \|\mathbf{x}_\ell - \mathbf{x}_i\|)\Big) \\ &+ \sum_{\{j:\|\boldsymbol{\xi}_j - \mathbf{x}_i\| < \delta_i\}} \sigma_i^j \frac{\boldsymbol{\xi}_j - \mathbf{x}_i}{\|\boldsymbol{\xi}_j - \mathbf{x}_i\|}\Big(\operatorname{sign}(\delta_i - \|\boldsymbol{\xi}_j - \mathbf{x}_i\|)\Big)\end{aligned}$$

where ℓ indicates the index of the leader agent.

Observe that $\Delta\mathbf{x}_i$ is controller agnostic, meaning that we can use this signal as the input to a PD or other controller that causes the agent to track toward the desired direction. This provides flexibility in applying the method to many different kinds of agents. In the simulations presented below, we assume a robot capable of omnidirectional travel, such as a quadcopter, and assume that the changes in direction are the commanded directions to the agent. The results shown in the demonstration below should generalize to other kinds of agents and controllers provided that the controller is able to quickly track the desired direction.

2.3 Collective Behaviors

The purpose of designing the interactions between individual agents and their environment is to produce useful behaviors for the swarm as a whole. We adopt a graph theoretic approach to describing and analyzing how the individual agent interactions produce a global behavior, following the example in [27]. Stated simply, viewing the swarm's interactions as a graph allows us to reason about these emergent behaviors. Each agent can be considered a node in the graph, while neighbouring agents' influence can be represented as directed connections between those nodes.

As noted in the introduction, there are two different graphs used in the work: an inter-agent influence graph, which is the graph used to compute the forces experienced by the agent in the previous section, and a communication graph, which we now describe. Both graphs use the same set of vertices, but their edge sets may differ. In this paper, we

assume that the two graphs are the same for simplicity, but future work should explore different topologies because detecting the set of nearest neighbors and the set of neighbors with whom an agent can communicate can be radically different. Consider, for example, a neighborhood topology obtained using visual sensing and a communication topology obtained using power-limited broadcast mechanisms; these two topologies would be very different.

2.3.1 Selecting Desired Inter-agent Distances

It is desirable to create collective behaviors that are robust to different physical scales and different numbers of agents. This presents a challenge for a group of agents that want to surround a building, because the spacing between agents depends on the size of the building and the number of agents.

In the spirit of distributed computing, we use a consensus algorithm for agents to reach an agreed up on ideal standoff distance d^*, yielding, for all agents i, $d_i \to d^*$. Each agent calculates the average distance of its neighbours and communicates this to each of the neighbours. It then increases its own value for d_i if the averages of its neighbours are larger, and decreases d_i if it is lower. This is precisely the Laplacian consensus algorithm described in [27, 34], where state is the average distance between an agent and its neighbors. This has the effect of allowing the agents to determine the ideal inter-agent distance without user input. It is well established that as long as the graph of agents is connected, all agents will eventually converge on the same value [27, 34].

2.3.2 Switching Between Modes

Using the communication topology, agents communicate commands they receive from a human operator to switch between *surround* and *travel* modes. This allows the human user to control the entire team while only communicating directly with a small number of agents. The algorithm by which this occurs has two parts, one part that applies to agents who receive direct commands from a human operator, and a second part for the agents with whom those

commanded agents interact. Consider first the agents that have been commanded to travel in a new direction. These agents switch to the *travel* mode and begin travelling in the given direction. These agents remain in *travel* mode until they have travelled the distance specified by the human operator. Upon reaching their destination or when the human cancels the command, they then change back into *surround* mode.

Consider now the agents that do not receive a direct command from the human. These agents watch their k selected neighbours. When an agent detects that one of its neighbours has changed into *travel* mode, it also changes into *travel* mode and sets the neighbour in *travel* mode to become its leader (if more than one neighbor is in *travel* mode, one is selected randomly). Agents remain in *travel* mode until they detect that their leader has changed back to *surround* mode.

In practice the two parts of this algorithm mean that the human induces a change to the entire group by selecting one or a few leaders and then letting the mode changes in those leaders propagate through the group. As long as the group stays connected, all agents will eventually switch to the *travel* mode and back to the *surround* mode.

2.3.3 Spanning Ring in Surround Mode

We are now in a position to precisely define the swarm-level behavior that we want to occur when the agents are in the different modes. We begin with *surround* mode.

The desired group configuration in *surround* mode is for the agents to distribute themselves around the perimeter of a building at equal inter-agent distances. This can be associated with a global property of the inter-agent graph. Formally, when the agent teams come to a stable position around a building, there should exist a subset of the agent graph's connections that form a *spanning ring* around the building. Using terminology from graph theory [16], we define a spanning ring as sub-graph that has a planar embedding (has no intersecting edges) which spans the entire graph (includes all agents) and such that each vertex has exactly two neighboring vertices. The existence of a spanning ring sub-graph

indicates that the agent team has successfully surrounded the building.

Importantly, the spanning ring is a global property that communicates the overall shape of the agents' formation to the human operator in an intuitive way. We discuss this further below.

2.3.4 Spanning Forest in Travel Mode

While the agent team is travelling between locations, each agent in the graph will be in *travel* mode. Except for the agents directly influenced by the human, each agent has identified exactly one leader agent to follow. Thus, the graph consists of multiple trees, one tree per agent under control of the human. Each tree is a *spanning tree* of the associated sub-graph of all agents connected to the same leader.

Thus, the global characteristic associated with the *travel* mode is a *spanning forest* sub-graph that consists of one or more spanning trees of agents. Metaphorically, this can be thought of as a "flock of flocks" or as the "vee of vee" that can be seen in nature and in WWII aerial tactics. Since all agents in the swarm are either leaders or are covered by exactly one spanning tree, the existence of the spanning forest indicates that the agent team has successfully adopted the *travel* mode.

As with the spanning ring, the spanning forest is a global property that communicates the overall shape of the agents' formation to the human operator in an intuitive way. We discuss this below.

2.4 Haptic Interface

The challenge for the human is to manage the team of autonomous agents effectively, without being overburdened or losing situational awareness through "heads-down" attention focused on a graphical user interface. In the present work, we use a visual display augmented with a haptic interface to represent the team and its environment to the user and to enable the user to command new locations and distributions to the swarm. The haptic element is accomplished

by creating virtual deformable volumes that enclose the spanning ring and spanning forest during the *surround* and *travel* modes, respectively. The user can feel, reposition, and deform these volumes, with corresponding changes in the location and shape of the team and in the distribution of the agents within the team. We present details of our approach in this section.

2.4.1 Modeling Clay: A Haptic Metaphor

Based on the results of a brainstorming breakout session at the 2012 AAAI Fall Symposium on Human Control of Biological Swarms, Diana et al. proposed the idea of using a deformable medium, such as modeling clay, as a "joystick" to command the distribution of large-scale swarm-like teams of homogeneous vehicles [13]. They demonstrated a molding scheme in which an operator formed modeling clay into various shapes in the view of an overhead camera and a team of micro robots replicated the formation commanded by the shaped clay. We modify the modeling clay metaphor so that a human can shape the distribution of agent teams by manipulating a *virtual* deformable volume through stretching, pulling and other operations. The modeling clay metaphor forms the basis for the haptic (force feedback) sensations that the user feels while distributing the agent team. Note that, unlike the work in [13], physical modeling clay is not used in our method; the concept and physics of modeling clay are used to generate the visual and haptic representation of the agent team.

We create a discrete approximation of the continuous clay metaphor by introducing potential force field spheres at the location of each of the agents in the spanning ring and at multiple points between agents. In essence, these potential spheres form the nodes of a "force graph" on which the haptic interaction forces and graphical representation of the deformable volume ("virtual modeling clay") are based.

Movement of the haptic interface in 3D space maps to movement of a graphical proxy on the display screen. The user exerts an external force, $\mathbf{f}_s$, on the network by moving the haptic proxy toward and into the outer boundary of one or more of its component spheres, as shown in Figure 2.1. The force $\mathbf{f}_s$ is calculated as a linear function of the penetration vector

$\boldsymbol{e}$, $\mathbf{f}_s = k_s\mathbf{e}$, where k_s is the stiffness constant of the sphere. Deformation of the network occurs under the action of the applied force $\mathbf{f}_s$. The haptic feedback force felt by the user in response to the interaction is then $\mathbf{f}_d = -\mathbf{f}_s$.

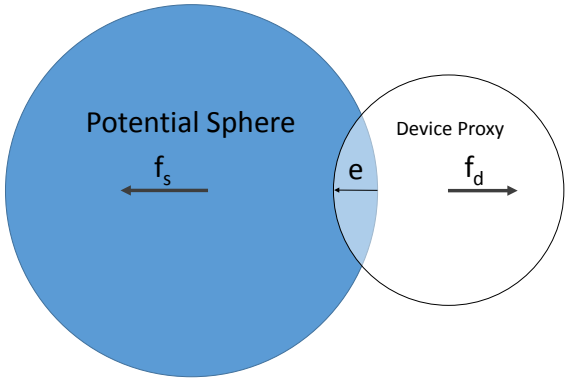

Figure 2.1: Haptic device proxy (white) in contact with a spherical potential force field (blue). $\mathbf{f}_s$ is the force the user exerts on the network of spheres and $\mathbf{f}_d$ is the return force felt by the user.

For force calculation purposes, the potential spheres also form the mass nodes of a virtual mass-spring-damper network. This dynamic network allows the model to update according to current team distribution and accept input forces from the operator to manipulate the team. To prevent the device proxy from passing between neighboring nodes, they are positioned a distance

$$\delta_{spacing} = r_n/n_{spacing}$$

apart, where r_n is the node radius and $4 \leq n_{spacing} \leq 6$. Choosing appropriate values for mass, spring, and damping constants allows each node to maintain sufficient distance relative to neighboring nodes, stabilize the model and create a distinguishable volume with which to interact.

The collective shape of the deformable volume should be a simple topology that the operator understands. To communicate a coherent group of agents in either *surround* or *travel* mode, a deformable ring is chosen as this topology, and is formed by connecting nodes that span across agent locations. The agent locations that define the deformable ring shape

are determined by the current mode, and will be discussed in the following section. The deformable ring forms the basis for computing the haptic feedback force felt by the operator.

2.4.2 Mapping of Agent Behaviors to Haptic Feedback

After the agent team has successfully surrounded a desired building or location in the environment, the operator may wish to explore the current state of the team. As mentioned above, a virtual deformable ring is created from the spanning ring formed by the agents. The resulting force graph provides the interaction forces for commanding and receiving feedback from the distributed agents. Placing additional nodes between agent locations ensures that the user can interact with a seemingly continuous deformable volume, rather than with discrete agents only.

While exploring the current distribution of the team, the operator receives force feedback when in contact with the volume. For a device proxy in contact with N nodes, the total haptic feedback force felt by the operator is

$$\mathbf{f}_h = \sum^{N} \mathbf{f}_d + \mathbf{f}_g,$$

where $\mathbf{f}_d$ represents the haptic force from a single node, as described previously, and $\mathbf{f}_g$ is a ground reaction force felt when the user comes in contact with the virtual ground. In addition to exploration, the user may also manipulate the distribution of the team. This is done by prodding the ring from the inside or outside, causing the volume to deform and propagate movement to each of the individual agents, as discussed in previous sections.

While the team is in motion from one location to another, the user may desire to feel the overall shape of the team while traveling as a spanning forest. For this purpose, a convex hull [12] of agent locations is computed based on their global positions in the environment. A *travel*-mode virtual deformable ring is formed by using the agents which are located on the edge of the convex hull as an ordered list of positions. Because the user would not benefit

from feeling the inside of the convex hull, we create a distinct enclosed deformable volume. When the device proxy comes within the 2-dimensional plane of nodal movement and is inside the convex hull, the user feels an out-of-plane (vertical) feedback force, $\mathbf{f}_c$, which creates a virtual surface to differentiate the team formation from the ground plane. The total haptic feedback force felt by the operator while in *travel* mode is

$$\mathbf{f}_h = \sum^{N} \mathbf{f}_d + \mathbf{f}_g + \mathbf{f}_c,$$

and represents the forces felt as the operator comes in contact with the set of nodes comprising the ring, the ground reaction force, and the convex hull force.

2.5 Simulation System and Demonstration

The graphical interface and haptic feedback of the agent team and its environment were programmed using CHAI 3D, "an open-source set of C++ libraries for computer haptics, visualization and interactive real-time simulation" (www.chai3d.org/concept.html). Figure 2.2 shows an operator's view of a simulation system that we have developed. The surrounded building has been chosen as a desired target by the user. From this *surround* state, the operator may manipulate the current distribution by interacting with the deformable ring (hidden from the user in this figure), or command the team to travel to one of the other buildings. A travel direction, $\mathbf{d}_t$, may be chosen by holding a button on the haptic device end effector. The operator holds the button down when close to a set of agents, drags the cursor in a specified direction and releases the button to commence movement. This feature allows the human operator to use global information to specify travel directions with respect to the current location of the team. While the button is held down, an addition travel force, $\mathbf{f}_t = -k_t \mathbf{d}_t$, is applied to the haptic device, where k_t is a stiffness parameter. This travel force serves to alert the user where the current location of the team is relative to the commanded location. The simulation system will be used to test proposed agent behaviors, feedback

algorithms and user interface capability as an integrated design.

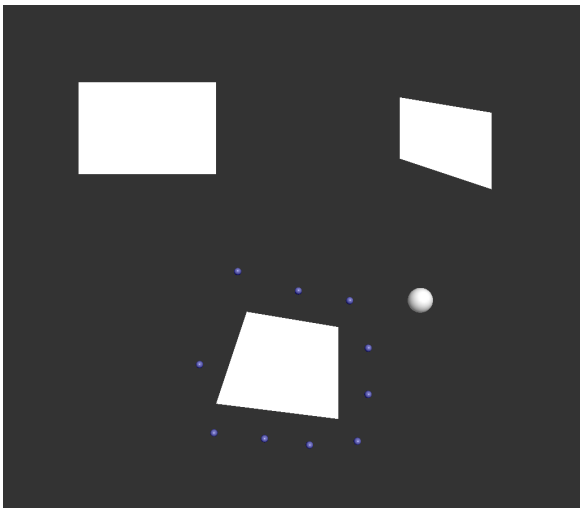

Figure 2.2: The human operator has global visibility of the team and environment from a bird's-eye view. A collection of buildings (white) are shown in the environment as candidate targets of interest. The team of agents (blue dots) are shown in *surround* mode around a building. The haptic proxy is the white sphere.

Figure 2.3 illustrates how user input via the haptic interface maps to commands to the autonomous team of agents. In this simulation there are 10 agents, although the methods presented in this paper are theoretically scalable to large numbers of agents. Empirically, values of $2 \leq k \leq 5$ tended to produce cohesive swarms and avoided fragmenting, where k is the number of other agents that an agent is influenced by, as described above. Results shown here are for $k = 5$. Sub-figures (a) and (b) show the team in *surround* mode around a building. The haptic proxy (white sphere) is not interacting with the team at this point in time. Sub-figure (a) shows the autonomous agents, whereas sub-figure (b) shows the deformable volume formed from the spanning ring computed from the agent distribution. In *surround* mode the operator may explore the state of the team and receive force feedback based on the current distribution. Sub-figures (c) and (d) show the operator changing the team distribution via the haptic proxy through stretching or pulling actions. When the operator manipulates the deformable volume, it influences the collective team behavior and individual agents adjust in a distributed way. Sub-figures (e) and (f) show the operator commanding the team to travel in a desired direction. During this movement, the operator

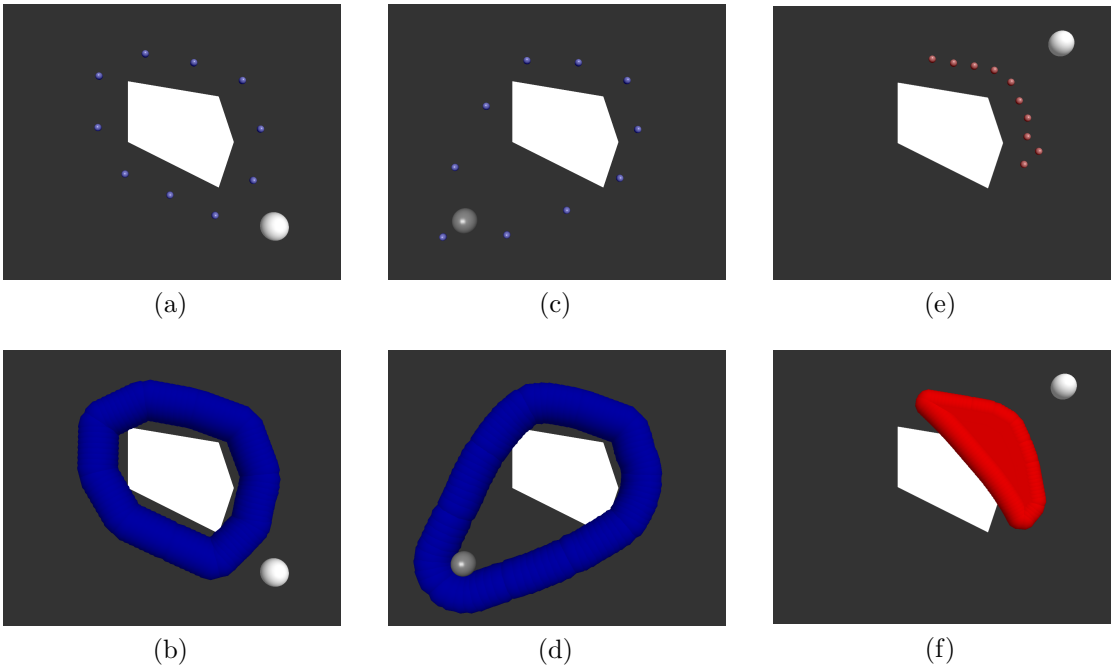

Figure 2.3: Demonstration of agent behaviors and haptic interactions. (a) The agent team autonomously surrounds a building geometry (shown in white). (b) A deformable surrogate is formed to intuitively sense the ring topology and team collective behavior. (c) External forces in the surrogate model propagate to movements by individual agents. (d) The human operator may manipulate the team distribution by pulling the volume in a desired direction. (e) The operator commands the team to travel in a desired direction. (f) The outer, convex hull shape is formed to provide haptic feedback based on tree topology.

may explore the overall shape of the spanning forest by interacting with the top virtual surface of the formation.

2.6 Summary and Future Work

In this work we have begun to develop methods to enable autonomous patrol and cordon by a distributed team of agents in an urban environment under the supervisory management of a single human user. We have developed *surround* and *travel* swarm behaviors that enable the team to autonomously form a cordon around a building and travel to a new location, respectively, where the building and new location can be selected by the user. Input and force feedback methods have been presented that allow the user to change and feel the distribution of the swarm via a haptic interface.

There are a number of open questions that have not been addressed in this demonstration. Many of these have been identified in the body of the paper, but three deserve to be emphasized in this section. First, although the demonstration showed that the algorithms controlling agent behavior tend to produce desirable team-level behavior, we have not presented an analysis of the conditions that guarantee these team-level behaviors. This problem is very important, because without understanding when the global behaviors are produced it is impossible to have bounds on how well the algorithms afford human interaction. Second, evidence needs to be gathered that the system works with real humans using real robots in real-world settings. A series of user studies and demonstrations with physical robots is needed to gain confidence that the system could be useful for the problems mentioned in the introduction. Third, the demonstration was only for a fixed number of agents. One of the desirable theoretical properties of a set of distributed agents is that their behavior is robust when some agents make errors and others are lost. The individual agent algorithms are designed to scale to more agents (e.g., using consensus to identify the desired inter-agent spacing) and the global properties exploited by the haptic controller should be invariant to the number of agents (e.g., the spanning ring and spanning forest), but evidence is needed to support this claim.

Chapter 3

Robust Behavior Design for a Team of Distributed Robots in Patrol and Search

C Kristopher Alder, Michael A Goodrich, and Mark B Colton. Robust behavior design for a team of distributed robots in patrol and search. In *International Conference on Intelligent Robots and Systems (IROS), 2016*. IEEE, 2016. — Under Review

3.1 Introduction

There is a lot of research being done on designing distributed algorithms that allow a team of robots to perform coordinated activities over wide spatial regions and periods of time. There are four main approaches to designing these algorithms: graph theoretic [27, 34], bio-inspired [5, 10], physics-based or physicomimetic [14, 38], and ad hoc. The literature in these areas is vast and rapidly growing so the citations above are only some examples of work in this area. There are many application areas of such work: emergency response [2, 24], search and rescue [21, 29], clean-up [22], and police and military applications [26].

An emerging challenge in these distributed systems is having a human manage them [42]. Theoretically, a single human or small team of humans should be able to manage a large distributed team if it is possible for the human to perceive the behavior of the team while it acts as a single unit [6, 9, 23, 32]. This can potentially tip the state-of-the-art from many human operators per robot [19, 28, 40] to many robots per human operator.

This paper presents ongoing work on developing a distributed algorithm that allows a human to manage a team of distributed robots tasked with performing patrol and cordon. From the set of approaches (graph theoretic, bio-inspired, etc.) we classify our approach as

ad hoc. We borrow our description of the task from our prior work [3] as follows: In a patrol and cordon scenario, an agent team works in an urban environment to search buildings. In *surround* mode the user selects a building or buildings for the team to surround, and the team autonomously forms the cordon. When the user selects a new building or buildings to surround, the team disengages from the original building, autonomously switches into a *travel* mode, and approaches and then re-forms around the new target.

The main contribution of this paper is in providing evidence that the ability of the team of robots to surround an object of interest is robust to many different types of perturbations: attrition of agents, movement of agents, and communication losses. The dynamics of the algorithms have some tricky non-linearities that make precise theoretical analysis difficult, so we present a series of empirical studies that provide evidence of robustness while giving insight into why the algorithms work.

3.2 Related Work

As the number of robots assigned to a task grows, it becomes increasingly difficult for a human operator to control each one individually [25]. Increasing the autonomy of the robots allows the users to control a larger number of robots or perform other tasks in addition to robot control.

One approach is to use agents that exhibit swarming behaviors. Swarming behaviors allow human operators to operate at a higher level of abstraction by specifying goals and priorities for the entire swarm [9, 20]. Individual robots then determine how to best reach the desired goals. This pattern of self-organization exists in nature, where complex flocking behaviors can form without centralized control [5, 10].

Current swarm models generally fall into one of three categories. The first, and simplest, category consists of swarms of homogeneous agents [10]. These agents are influenced only by other agents and have no internal memory or state. The second category of swarming behaviors adds an internal state machine to each agent [30]. The behavior of individual

agents or the swarm as a whole depends on the agents's states. Transitions between different states are governed by interactions between the agents as well as other objects in the world. The third, and most sophisticated, group of models consists of groups of heterogeneous agents. In these models, several different classes of agents exist that exhibit different individual behaviors. Often, this is a result of agents being specialized to perform specific subtasks. Existing research into this category has focused on a subset of agents being directly influenced by human operators, while others are only influenced by humans indirectly [8].

An important feature of most swarm models is decentralization. A specific objective of decentralized systems is minimizing the amount of global information [27]. This allows the individual agents to act more autonomously and makes the swarm more reliable. In a real-world scenario, communication between robots will often be wireless and prone to disconnections, interference, and other noise. Additionally, communication bandwidth and range are often limited. Collecting information from each robot and then disseminating that information back to the swarm can be slow and difficult. Making the system less dependent on a single global source of information allows for improved robustness [41]. It also has the benefit of allowing the swarm to adapt as individual agents are disabled due to hardware or software failures.

Since no one agent is responsible for the state of all other agents, it is possible for one or more agents to disconnect from the swarm. This tendency for swarms to divide or lose agents is well-documented. [10, 22] Several methods for measuring the connectivity and cohesion of swarms exist [15, 41].

3.3 Individual Behaviors

This section presents the ad hoc model for performing cordon and patrol and is an adapted version of what was first presented in [3]. It is necessary to include this review so that the empirical results can be understood.

We restrict attention to agents that can sense only two types of objects in the world:

other agents and obstacles/points of interest. Based on sensor readings from these types of objects, each agent computes a force that pushes or pulls it in a direction that (a) keeps it close (but not too close) to its neighbors and (b) either keeps it in proximity to the boundary of an object if the agent is in a *surround* mode or helps it avoid an object if the agent is in *travel* mode.

3.3.1 Nearest-Neighbour Selection

We adopt a nearest neighbor approach (in contrast to metric-based methods [10, 35]) to determine which agents influence which other agents because prior work suggests that it produces collectives that are less likely to fragment in the presence of large perturbations [5, 7, 18]. This means that the forces that an agent experiences are determined by a limited number of agents. This induces what we call an *interagent influence topology*. Later in this paper, we will consider a different topology, which we call the *communication topology*, that may differ from the interagent topology.

Let k denote the number of other agents that an agent is influenced by. Let $\mathbf{x}_i$ denote the position of the i^{th} agent in the collective and let $N(\mathbf{x}_i; k)$ denote the positions of the k-nearest neighbors of agent i. From $\mathbf{x}_i$ and $N(\mathbf{x}_i; k)$ distances between an agent and each of its neighbors are easily computed. Let $\mathbf{X}_i$ denote the matrix constructed from $N(\mathbf{x}_i; k)$ by sorting the neighbor set from nearest to farthest from the agent. Thus,

$$\mathbf{X}_i = [\mathbf{x}_1 \mathbf{x}_2 \dots \mathbf{x}_k] \tag{3.1}$$

where we assume that each $\mathbf{x}_i$ is a column vector and $\mathbf{x}_1$ is the neighbor nearest to agent i, $\mathbf{x}_2$ is the next closest, and so on.

3.3.2 Obstacle Sensing

Assume a radar or lidar-type sensor that is able to measure both distance to and direction of objects in the world. Assume further that the radar sends out "pings" at regular intervals in a 360-degree arc. Larger objects, such as walls, will be detected by multiple of these simulated radar "pings". This yields a set of readings of obstacle locations uniformly distributed around agent i, $\{\boldsymbol{\xi}_i^j : j \in \{1, 2, \ldots, \frac{360}{\Delta} - 1\}\}$ where Δ is the angle between the different pings. Note that we are using a convention where the symbol for agent-related parameters uses a letter from the English alphabet while the symbol for the corresponding object-related parameter uses the corresponding letter from the Greek alphabet.

We sort these readings from nearest to farthest and drop readings that are beyond the range of the sensor. This sorted vector of readings is denoted by the array $\boldsymbol{\Xi}_i$. Because objects outside of the sensors' range may not be detected at all, the size of the array may vary as agents move in the world and objects enter and leave sensor range. Thus,

$$\boldsymbol{\Xi}_i = \left[\boldsymbol{\xi}_i^1 \boldsymbol{\xi}_i^2 \ldots \boldsymbol{\xi}_i^n\right] \tag{3.2}$$

where $n \leq \frac{360}{\Delta}$ is the number of sensor readings within sensor range and $\boldsymbol{\xi}_i^1$ is the location of the sensor ping closest to agent i, $\boldsymbol{\xi}_i^2$ the next closest reading, and so on.

3.3.3 Emphasizing Unique Influences

The algorithm limits the amount of influence a small cluster of agents or sensor pings can exert; agents that lie in a very similar direction to other neighbouring agents are given lower influence. For each agent $a \in N(\mathbf{x}_i, k)$, assign a weighting w_i^a that is a function of how unique agent a's influence is relative to other agents. For the nearest agent, which is identified by $a = 1$ since we sorted agents from nearest to farthest, let $w_i^1 = \pi$. This initializes the

weight-assignment algorithm. For each other agent $a \in \{2, \ldots, k\}$, let w_i^a be given by

$$w_i^a = \min_{1 \leq b < a} \cos^{-1}\left(\frac{\mathbf{x}_a^T \mathbf{x}_b}{\|\mathbf{x}_a\| \|\mathbf{x}_b\|}\right) \tag{3.3}$$

This equation iteratively assigns a weight to every agent $a \in \{2, \ldots, k\}$ proportional to the smallest angle between agent a and every other agent that has previously been assigned a weight. The normalized inner product gives the cosine of the angle between unassigned agent a and every previously assigned agent b. The arc cosine gives the angle between agent a and agent b, and the smallest such angle indicates how directionally similar that agent is to every previously assigned agent. If there are other, closer agents in the general direction of agent a then agent a is in a cluster of agents and shouldn't receive a high weighting.

The weights w_i are then normalized so that they sum to 1.

The same algorithm can be applied to each column in $\boldsymbol{\xi}_i$ yielding a set of "uniqueness" weights for each ping, ω_i^j where $j \in \{1, 2, \ldots, n\}$.

The weights given by the algorithm described in the previous section determine *how much* an agent in agent i's neighborhood and how much obstacles will influence agent i. We now need to determine *how* agent i's behavior should be influenced by those agents. Depending on mode, agents want to be close but not too close to other agents and to objects in the world. Let d_i and δ_i denote the ideal standoff distance that agent i wants to maintain between agents and obstacles, respectively.

If a neighbor of agent i is too far away then agent i should be attracted to that neighbor. Formally, agent i is attracted to neighbor j when $\|\mathbf{x}_i - \mathbf{x}_j\| > d_i$. The strength of the attractive influence grow as the distance increases. Similarly, agents that are closer than the stand-off distance d_i exert a repulsive force (to prevent collisions) that approaches infinity as the distance between the two agents approaches zero. The strength of the force

from agent j to agent i is given by

$$s_i^j = w_i^j \Big| \frac{1}{d_i} - \frac{1}{\|\mathbf{x}_i - \mathbf{x}_j\|} \Big| \tag{3.4}$$

The radar "pings" represented by Ξ_i exert similar repulsive and attractive forces on agent i with strengths scaled by ω_i. This yields the strength of the force from sensor reading j on agent i as

$$\sigma_i^j = \omega_i^j \Big| \frac{1}{\delta_i} - \frac{1}{\|\mathbf{x}_i - \boldsymbol{\xi}_j\|} \Big| \tag{3.5}$$

3.3.4 Modes of Operation

The direction that the agent should travel is given by the sum of the forces derived from other agents and from objects in the world. Agents have two different modes or states: *surround* and *travel*; some forces are ignored in some of the modes. In *surround* mode the forces on the agent are given by:

$$\begin{aligned} \Delta\mathbf{x}_i &= \sum_{j=1}^{k} s_i^j \frac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|} \Big(\text{sign}(d_i - \|\mathbf{x}_j \mathbf{x}_i\|) \Big) \\ &+ \sum_{j=1}^{n} \sigma_i^j \frac{\boldsymbol{\xi}_j - \mathbf{x}_i}{\|\boldsymbol{\xi}_j - \mathbf{x}_i\|} \Big(\text{sign}(\delta_i - \|\boldsymbol{\xi}_j - \mathbf{x}_i\|) \Big). \end{aligned} \tag{3.6}$$

This equation sums over all neighbors (the first line) and all objects (the second line). The vectors point in the direction of neighbors/objects, and the sign of these vectors is determined by whether the objects/neighbors attract or repel the agent.

In *travel* mode, two changes are made to this equation: First, only one other agent is considered, a special agent called the leader. The leader agent is denoted with the index ℓ.

Second, the agent is never attracted to an object, only repelled. This yields

$$\begin{aligned}\Delta\mathbf{x}_i &= s_i^{\ell}\frac{\mathbf{x}_\ell - \mathbf{x}_i}{\|\mathbf{x}_\ell - \mathbf{x}_i\|}\Big(\operatorname{sign}(d_i - \|\mathbf{x}_\ell - \mathbf{x}_i\|)\Big) \\ &+ \sum_{\{j:\|\boldsymbol{\xi}_j - \mathbf{x}_i\| < \delta_i\}} \sigma_i^{j}\frac{\boldsymbol{\xi}_j - \mathbf{x}_i}{\|\boldsymbol{\xi}_j - \mathbf{x}_i\|}\Big(\operatorname{sign}(\delta_i - \|\boldsymbol{\xi}_j - \mathbf{x}_i\|)\Big)\end{aligned}$$

The "root" leader uses a variant of the shared controller from [11] that takes a direction vector (from a human) to a target location and is repelled by obstacles.

Observe that $\Delta\mathbf{x}_i$ is controller agnostic, meaning that we can use this signal as the input to a PD or other controller that causes the agent to track toward the desired direction. In the simulations presented below, we assume a robot capable of omnidirectional travel and assume that the changes in direction are the commanded directions to the agent.

3.4 Collective Behaviors

3.4.1 Spanning Ring in Surround Mode

Agents in *surround* mode should distribute themselves around the perimeter of a building at equal inter-agent distances. Again, using the definition from [3], when the agents come to a stable position around a building, there should exist a subset of the agent graph's connections that form a *spanning ring* around the building. A *spanning ring* as sub-graph that has a planar embedding (has no intersecting edges) which spans the entire graph (includes all agents) and such that each vertex has exactly two neighboring vertices [16].

3.4.2 Spanning Forest in Travel Mode

Again, using the definition from [3], except for the agents directly influenced by the human, each agent has identified exactly one leader agent to follow. Thus, the graph consists of multiple trees, one tree per agent under control of the human operator. Each tree is a *spanning tree* of the associated sub-graph of all agents connected to the same leader. Thus,

the global characteristic associated with the *travel* mode is a *spanning forest* sub-graph that consists of one or more spanning trees of agents.

3.4.3 Switching Between Modes

Again, this section uses a description adapted from [3], Using a communication topology, agents communicate commands they receive from a human operator to switch between *surround* and *travel* modes. The human communicates with only a small number of agents. When the human tells this subset to travel in a new direction, they switch *travel* mode and begin travelling in the given direction. These agents remain in *travel* mode until they have travelled the distance specified by the human operator. Upon reaching their destination or when the human cancels the command, they then change back into *surround* mode.

Agents that do not receive a direct command from the human watch their k selected neighbours. When an agent detects that one of its neighbours has changed into *travel* mode, it also changes into *travel* mode and sets the neighbour in *travel* mode to become its leader. If more than one neighbor is in *travel* mode, one is selected randomly. Agents remain in *travel* mode until they detect that their leader has changed back to *surround* mode.

3.4.4 Selecting Desired Inter-Agent Distances

Robustness of the surround mode algorithm requires that it work for differently sized objects. This requires a way for agents to spread themselves more thinly around the outside of large object and more compactly around a small object. Agents use a consensus algorithm [34] to select ideal standoff distance d^* such that, for all agents i, $d_i \to d^*$. Each agent calculates the average distance of its neighbours,

$$\bar{d_i} = \frac{1}{k} \sum_{j=0}^{k} ||\mathbf{x}_i - \mathbf{x}_j||$$

and communicates this to each of the neighbours.

An agent increases d_i if the averages of its neighbours are larger than d^*, and decreases d_i if it is lower, as given by

$$\Delta d_i \propto \frac{1}{k} \sum_{j=0}^{k} \bar{d}_j$$

. This is precisely the Laplacian consensus algorithm described in [27, 34], where state is the average distance between an agent and its neighbors. This has the effect of allowing the agents to determine the ideal inter-agent distance without user input. Large objects push agents away from the boundary resulting in larger interagent distances.

3.5 Spanning Ring is Locally Stable

The spanning ring that agents form while in *surround* mode is a locally stable equilibrium. A direct proof of this is difficult because of nonlinearities in the way agents select neighbours as well as how the influence weights are assigned for both agents and obstacle detections. These nonlinearities are key to the swarms' ability to surround non-convex obstacles, especially the distribution of weights to agents and obstacles at unique directions from the agents; subjectively, the way of assigning weights creates a dispersion that allows the agent to surround obstacles when they emerge from *travel* mode. We provide empirical evidence that once agents have formed a spanning ring around an obstacle they will return to a ring formation when perturbed. We give three increasingly convincing demonstrations of this behavior.

First we allow the swarm to come to a spanning ring solution around a point obstacle. Once the swarm has reached a steady state, the positions of all but one agent are fixed. This maintains the positions, relative distances, and neighbour selection of each of the agents. The remaining agent is then displaced in a random direction. The agent quickly resumes its original position around the obstacle. A plot of the path of the perturbed agent, as well as that of the fixed-position agents, is given in Figure 3.1.

The demonstration can be repeated without holding the positions of all non-perturbed

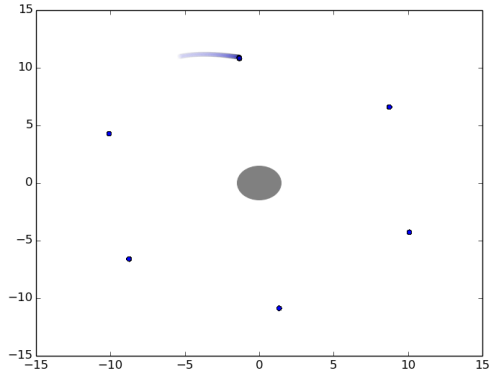

Figure 3.1: Perturbing A Single Agent While Holding Others Constant

agents constant. In this case, the non-perturbed agents move in response to one agent's position changing, but all quickly return to an evenly-spaced distribution around the point obstacle, albeit in slightly different locations. The results of one such simulation are given in Figure 3.2.

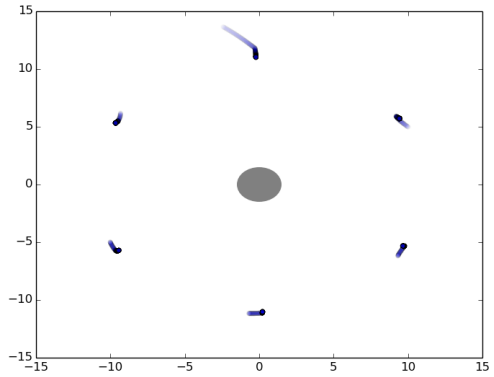

Figure 3.2: Perturbing A Single Agent While Allowing Free Motion

Finally, it can be shown that perturbing all agents in a random direction does not prevent them from quickly returning to a spanning ring solution. This is shown in Figure 3.3.

These demonstrations suggest that the spanning ring is a locally stable equilibrium of the underlying dynamics and that agents displaced within the nearby region will return to the equilibrium.

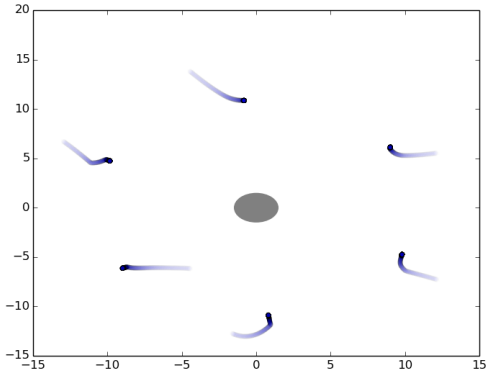

Figure 3.3: Perturbing A Single Agent While Allowing Free Motion

3.6 Measuring Swarm Robustness

In the previous section, we made a case that a spanning ring was a locally stable equilibrium of the algorithm dynamics. In this section, we address the question of "how big" this locally stable equilibrium is. Stated another way, we conduct some experiments that indirectly estimate the extent to which the swarm's parameters can change while still producing the desired behavior. This is very useful because the size of the basin of attraction characterizes how robust the spanning ring equilibrium can be; large basins are likely to be more robust to perturbations than small basins.

We tested swarms using these dynamics across a variety of scenarios containing different polygonal or point obstacles. The ten scenarios we designed contain cases with one or two polygons, both convex and concave, as well as scenarios using either one or two point obstacles. To measure the sensitivity of the system to its initial parameters, simulations were run over each of the scenarios varying (a) the number of agents in the system and (b) the number of inter-agent connections made by each agent. Then, the robustness of the system was measured by measuring the swarms' ability to respond to agent attrition and communication disruptions.

3.6.1 Varying the Number of Agents

First, the number of agents in the swarm was varied. Keeping the number of inter-agent connections set to 4, a group of agents was placed at randomized starting locations in the world within detection distance of the obstacles. The simulation was then run until the agents formed and maintained a spanning ring for 1,000 continuous iterations and have minimal change in position. If the simulation ran for 100,000 iterations without meeting these conditions, the swarm was considered to have failed to converge. Teams of $\{4, 6, \ldots 14\}$ agents were tested. Each combination of team size and scenario were run fifty times. Results are given in Table 3.1.

	Agents					
	4	6	8	10	12	14
Single Point	100	100	100	100	40	92
Multiple Points	100	100	100	100	94	34
One Convex Reg	100	90	98	64	92	50
Multi Convex Reg	100	100	100	100	96	92
One Convex Irreg	100	100	100	100	100	100
Multi Convex Irreg	100	22	4	94	22	98
One Concave Reg	50	100	100	68	100	84
Multi Concave Reg	100	66	98	66	96	66
One Concave Irreg	100	100	100	94	54	74
Multi Concave Irreg	100	100	34	0	100	92

Table 3.1: Percentage Reaching Convergence with Given Number of Agents

Note that even in situations where the swarm did not converge into a spanning ring solution, agents still reached stable positions and successfully distributed themselves around the obstacles in the world. This is evidence that the spanning ring is locally stable but other distributions are also locally stable. For example, the scatter plots in Figures 3.4 and 3.5 show the locations of fourteen agents in the *Multiple Points* scenario and eight agents in the *Multiple Convex Irregular* scenario over a period of several hundred iterations.

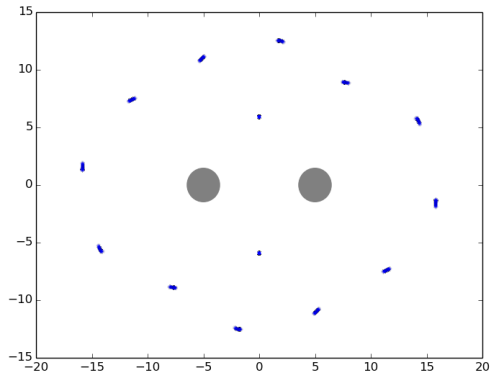

Figure 3.4: 14 Agents around Multiple Point Obstacles

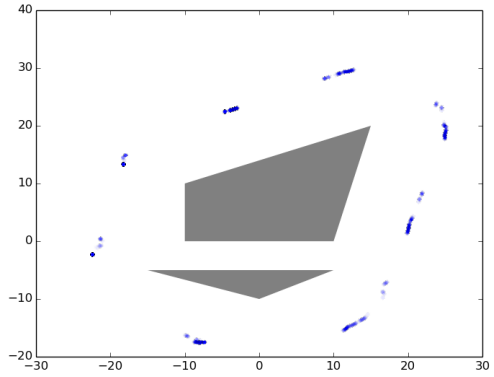

Figure 3.5: 8 Agents around Multiple Convex Irregular Obstacles

3.6.2 Varying the Number of Inter-Agent Connections

Next, effects of different numbers of inter-agent connections were tested for $k \in \{2, 3, 4, 5\}$. In each scenario the number of agents was fixed at 8. As before, each simulation was allowed to run until either the swarm converged and formed a ring or 100,000 iterations passed without success. Each value and scenario was run fifty times. The results are given in Table 3.2.

A key finding here is that swarms with small numbers of connections exhibited many more problems surrounding obstacles. Swarms with only two connections per agent were unlikely to find a solution in many circumstances. However, even well-connected swarms were ineffective in some of the scenarios. Nevertheless, when agents have relatively large

	k			
	2	3	4	5
Single Point	100	82	100	100
Multiple Points	100	100	100	100
One Convex Reg	100	82	98	98
Multi Convex Reg	12	98	100	100
One Convex Irreg	0	98	100	100
Multi Convex Irreg	0	46	4	0
Single Concave Reg	0	0	100	2
One Concave Reg	0	100	98	100
One Concave Irreg	78	92	100	100
Multi Concave Irreg	0	0	34	75

Table 3.2: Percentage Reaching Convergence with Given Number of Inter-Agent Connections

neighborhoods they are much more likely to yield spanning rings; the basin of attraction for the algorithms tend to grow as neighborhood sizes increase.

3.6.3 Robustness with Agent Attrition

To measure the robustness of the system, simulations were run similar to those described above. A team of eight agents, each connecting with up to four other agents, were used. Scenarios with low completion rates were removed because they appear to not have large regions of attraction for the desired spanning ring. Once the agents converged on a solution, there was a probabilistic chance that each could go offline. Agents that have gone offline are removed from the system and no longer influence other agents or have any other effect. The remaining agents are then allowed to adapt and re-form a spanning ring around the obstacle or obstacles. 10%, 20%, 30%, 40% and 50% attrition rates were used. Fifty simulations were run with each level of attrition. Due to the dependence on agents coming to an initial solution, only those scenarios with success rates greater than 90% for teams of 8 agents with 4 connections were used for these tests. The results are given in Table 3.3.

In almost all cases the teams were able to adapt to the sudden disappearance of agents. This provides further evidence that the system works effectively with differing numbers of agents and that, once found, a spanning ring equilibrium is likely to be relatively stable in the presence of perturbations.

	Attrition				
	10%	20%	30%	40%	50%
Single Point	100	100	100	100	100
Multiple Points	100	100	100	100	100
One Convex Reg	100	100	100	100	100
Multi Convex Reg	100	100	100	100	100
One Convex Irreg	100	100	100	100	100
Multi Concave Reg	92	82	78	84	96
Single Concave Irreg	82	84	84	94	98

Table 3.3: Percentage Reaching Re-Convergence with Given Attrition Rate

3.6.4 Robustness with Unreliable Communication

The final test we performed was to measure the amount of robustness of the swarm with unreliable communication. We use a very simplified model of wireless communication, where at each iteration there is a chance that a each agent will be unable to communicate with each of its neighbouring agents. If an agent is unable to communicate with an agent that would otherwise be selected as a neighbour, it will select the next nearest agent it can communicate with instead.

To show that the swarm is still likely to find a solution around a given obstacle or obstacles, the positions of each agent at each iteration were recorded and plotted similar to those in Figures 3.4 and 3.5. Measurements are only taken after agents have been in a single location with only small movements for over 100 iterations to avoid noise from the agents' initial approach to the obstacle. Visual inspection of the plotted data shows that the agents do in fact come to a spanning ring around the obstacle as expected. Two typical plots are given here. Figure 3.6 shows six agents surrounding a single point obstacle located at the origin. Figure 3.7 shows eight agents surrounding a U-shaped polygonal obstacle from the *Single Concave Regular* scenario. Observe the drift of the agents in the first case but the relatively constant inter-agent spacings. This suggests that the spanning ring is robust to the presence of communication drop-outs even though agents drift.

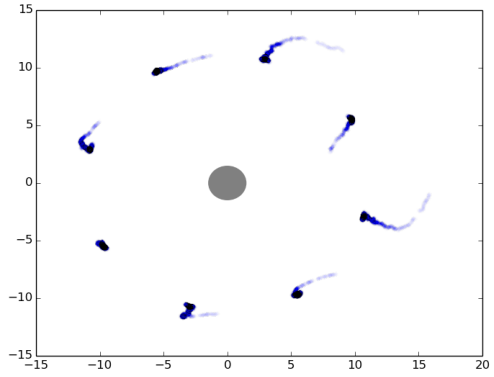

Figure 3.6: Agents Surrounding a Point Obstacle

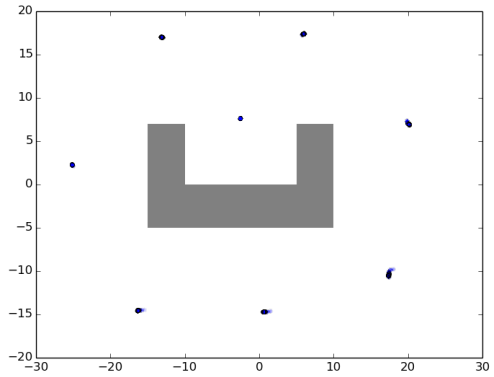

Figure 3.7: Agents Surrounding a Concave Polygonal Obstacle

3.7 Conclusion

We have presented a distributed model for enabling swarms of robots to carry out cordon and search tasks. The model uses ideas from graph theory to represent the interactions between robotic agents in the system. The presence of a spanning ring or spanning forest topology during these tasks represent the main behaviors of the team. We demonstrate empirically that the spanning ring topology is a locally stable equilibrium. The dynamics of the system allow it to function in a variety of situations and make it robust in the presence of communication errors or agent attrition.

Chapter 4

Effect of Diffusion on Cordon

4.1 Introduction

A process was described in Section 3.3.3 that each agent uses to determine the amount of influence of each neighbouring agent or sensor reading has. Weighting influences in this way causes the swarms' behaviors to have a diffusing effect. This allows the swarm to spread out and prevents the agents from clustering together in small clumps. It also introduces a complicated non-linear factor into the swarming algorithm. This nonlinearity makes a formal analysis of the swarm's behavior difficult.

Here we demonstrate empirically that this diffusion factor is essential for swarms to converge in a spanning ring. Earlier, in Section 3.6 a series of experiments were performed to measure the swarm's ability to form a cordon under a variety of conditions. To measure the effect of the diffusion factor, we carried out similar experiments with the diffusion factor disabled. This has the effect of making all inter-agent and sensor readings have equal strength.

First we replicated the experiments of Section 3.6.1 that measured the swarm's ability to form a cordon with different numbers of agents. Then, we measured how changing the number of inter-agent connections effects the swarm's performance in the cordon task. This was initially done in Section 3.6.2. The results of both of these experiments are given and discussed in the following sections.

4.2 Varying the Number of Agents

As performed earlier, simulations were run on ten different configurations of simulated buildings. The same configurations used earlier were used here to allow direct comparision of results. The number of agents in each team was varied across values of $\{4, 6, \ldots 14\}$. As before, the number of inter-agent connections was held constant at 4. For each combination of team size and scenario, fifty separate simulations were run with randomized initial agent positions. If the team was able to form a spanning ring for 1,000 continuous iterations without significant change in position it was considered to have succeeded. If 100,000 iterations passed without success, the team is considered to have failed.

Rates for completion are given in Table 4.1. Comparing these results to those given in Table 3.1 shows that the team fails to find a solution except in the very simplest scenarios or when the team size is very small, having only four or six agents. Considering that each agent connects to as many as four others, the agent graph for small teams (4–6 agents) are likely to be fully- or nearly fully-connected. In a fully-connected graph a spanning ring will certainly exist. It is quite likely that the high completion rates for smaller team sizes may not actually correspond to teams surrounding the building and instead reaching stable positions on one side of the building.

	Agents					
	4	6	8	10	12	14
Single Point	100	100	100	100	100	100
Multiple Points	100	100	100	100	84	98
One Convex Reg	98	16	70	20	0	0
Multi Convex Reg	100	64	0	0	0	0
One Convex Irreg	100	14	0	0	0	0
Multi Convex Irreg	100	0	0	0	0	0
One Concave Reg	100	0	0	0	0	0
Multi Concave Reg	100	100	0	0	0	0
One Concave Irreg	100	18	2	0	0	0
Multi Concave Irreg	100	74	0	0	0	0

Table 4.1: Percentage Reaching Convergence with Given Number of Agents

4.3 Varying the Number of Inter-Agent Connections

Next, we tested the effects of varying the number of inter-agent connections. Values of $k \in \{2, 3, 4, 5\}$ were used. The same ten building configuration as used elsewhere were used, and team size was kept constant at 8. The criteria for considering a cordon a success are the same as used in the previous section as well as elsewhere in this thesis.

The results of these trials are given in Table 4.2. The team was likely to converge on a solution only in the most simple scenarios ('Single Point' and 'Multiple Points'). The slight chances for convergence for the 'One Convex Regular' and 'One Concave Irregular' scenarios at k-values of 4 and 5, respectively, are notable. These can be explained similarly to simulations with high success reported in the previous section. Highly connected inter-agent graphs, such as form with high k-values, are much more likely to have spanning ring sub-graphs regardless of the teams' ability to surround a building.

	k			
	2	3	4	5
Single Point	100	100	100	100
Multiple Points	56	100	100	100
One Convex Reg	2	0	70	0
Multi Convex Reg	0	0	0	0
One Convex Irreg	0	0	0	0
Multi Convex Irreg	0	0	0	0
Single Concave Reg	0	0	0	0
One Concave Reg	0	0	0	0
One Concave Irreg	0	0	2	34
Multi Concave Irreg	0	0	0	0

Table 4.2: Percentage Reaching Convergence with Given Number of Inter-Agent Connections

Chapter 5

Summary and Future Work

5.1 Summary

This thesis presents a solution for human-robots teams to coordinate in cordon-and-search tasks. This includes several important contributions. A group of desirable collective behaviors for a robot team have been defined using concepts from graph theory. The behaviors of individual agents in the team have also been defined, as well as specifying how robots in the team should communicate with each other.

Simulations of agents with these behaviors were carried out with a variety of initial conditions, showing that the algorithm succeeds in a range of conditions. These results demonstrate that the algorithm functions with teams of many different sizes, levels of inter-agent communication, and initial location. The simulated robot teams were able to carry out cordon-and-patrol tasks even with significant levels of agent attrition and unreliable communications. Finally, the unique approach for introducing diffusion into the swarm has been shown to be critical to the teams' success for the cordon operation.

5.2 Future Work

There are several areas for future work on this problem. While the algorithm was designed to enable humans to easily manage the robot swarm, the effectiveness and usability of this system needs to be evaluated. Some initial work has been carried out examining the use of haptic interfaces to control robot teams using this system as published in Chapter 2, but much more needs to be done.

The algorithms could be extended to allow human users to manipulate the shape or distribution of the robot team after it has formed a cordon autonomously. This would allow human users to improve the swarm's performance using experience, intuition, or other information not available to the robots.

Finally, although the diffusion algorithm given is essential to the system's performance, it remains difficult to validate from a theoretical standpoint. Ideally a proof would be found showing that under the algorithm a graph with a spanning ring is a stable equilibrium. It may also be possible to replace the diffusion algorithm with another, simpler method that would have a similar effect and be easier to prove the correctness of.

References

[1] C. R. Twomey A. Strandburg-Peshkin. Visual sensory networks and effective information transfer in animal groups. *Current Biology*, 23(17):R709–R711, 2013.

[2] L. Alboul, J. Saez-Pons, and J. Penders. Mixed human-robot team navigation in the GUARDIANS project. In *Proccedings of the International Conference on Safety, Security, and Rescue Robotics*, Sendei, Japan, October 2008.

[3] C Kristopher Alder, Samuel J McDonald, Mark B Colton, and Michael A Goodrich. Toward haptic-based management of small swarms in cordon and patrol. In *Swarm/Human Blended Intelligence Workshop (SHBI), 2015*, pages 1–8. IEEE, 2015.

[4] R. C. Arkin. *Behavior-Based Robotics*. MIT Press, Cambridge, Massachusetts, 1998.

[5] Michele Ballerini, Nicola Cabibbo, Raphael Candelier, Andrea Cavagna, Evaristo Cisbani, Irene Giardina, Vivien Lecomte, Alberto Orlandi, Giorgio Parisi, Andrea Procaccini, et al. Interaction ruling animal collective behavior depends on topological rather than metric distance: Evidence from a field study. *Proceedings of the National Academy of Sciences*, 105(4):1232–1237, 2008.

[6] Daniel S Brown and Michael A Goodrich. Limited bandwidth recognition of collective behaviors in bio-inspired swarms. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems*, pages 405–412. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[7] Daniel S Brown, Shin-Young Jung, Michael Goodrich, et al. Balancing human and inter-agent influences for shared control of bio-inspired collectives. In *Proceedings of the 2014 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 4123–4128. IEEE, 2014.

[8] Daniel S Brown, Sean C Kerman, and Michael A Goodrich. Human-swarm interactions based on managing attractors. In *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*, pages 90–97. ACM, 2014.

[9] Daniel S Brown, Michael A Goodrich, Shin-Young Jung, and Sean Kerman. Two invariants of human-swarm interaction. *Journal of Human-Robot Interaction*, 1(1):78–95, 2016. To appear.

[10] I. D. Couzin, J. Krause, R. James, G. D. Ruxton, and H. R. Franks. Collective memory and spatial sorting in animal groups. *Journal of Theoretical Biology*, 218(1), September 2002.

[11] J. W. Crandall and M. A. Goodrich. Experiments in adjustable autonomy. In *Proceedings of the 2001 IEEE International Conference on Systems, Man and Cybernetics*, October 2001.

[12] M. de Berg, M. van Kreveld, Mark Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer, 2000.

[13] Matteo Diana, Jean Pierre De La Croix, and Magnus Egerstedt. Deformable-medium affordances for interacting with multi-robot systems. *IEEE International Conference on Intelligent Robots and Systems*, pages 5252–5257, 2013. ISSN 21530858. doi: 10.1109/IROS.2013.6697116.

[14] M. R. D'Orsogna, Y. L. Chuang, A. L. Bertozzi, and L. S. Chayes. Self-propelled particles with soft-core interactions: Patterns, stability, and collapse. *Phys. Rev. Lett.*, 96:104302, Mar 2006. doi: 10.1103/PhysRevLett.96.104302. URL `http://link.aps.org/doi/10.1103/PhysRevLett.96.104302`.

[15] Veysel Gazi and Kevin M Passino. A class of attractions/repulsion functions for stable swarm aggregations. *International Journal of Control*, 77(18):1567–1579, 2004.

[16] Chris Godsil and Gordon F Royle. *Algebraic graph theory*, volume 207. Springer Science & Business Media, 2013.

[17] M. A. Goodrich, B. S. Morse, D. Gerhardt, J. L. Cooper, M. Quigley, J. A. Adams, and C. Humphrey. Supporting wilderness search and rescue using a camera-equipped mini UAV. *Journal of Field Robotics*, 25(1-2):89–110, 2008.

[18] M. A. Goodrich, P.B. Sujit, S. Kerman, B. Pendleton, and J. Pinto. Enabling human interaction with bio-inspired robot teams: Topologies, leaders, predators, and stakeholders. Technical Report BYU-HCMI Technical Report 2011-1, Brigham Young University, 2011.

[19] Michael A. Goodrich. On maximizing fan-out: Towards controlling multiple unmanned vehicles. In M. Barnes and F. Jentsch, editors, *Human-Robot Interactions in Future Military Operations*. Ashgate Publishing, Surrey, England, 2010.

[20] Michael A Goodrich, Sean Kerman, and Shin-Young Jun. On leadership and influence in human-swarm interaction. In *AAAI Fall Symposium: Human Control of Bioinspired Swarms*, 2012.

[21] Hank Jones and Pamela Hinds. Extreme work teams: Using swat teams as a model for coordinating distributed robots. In *Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work*, CSCW '02, pages 372–381, New York, NY, USA, 2002. ACM. ISBN 1-58113-560-2. doi: 10.1145/587078.587130. URL `http://doi.acm.org/10.1145/587078.587130`.

[22] S.Y. Jung and M. A. Goodrich. Shaping couzin-like torus swarms through coordinated mediation. In *Proceedings of the 2013 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1834–1839. IEEE, 2013.

[23] Andreas Kolling, Katia Sycara, Steven Nunnally, and Michael Lewis. Human swarm interaction: An experimental study of two types of interaction with foraging swarms. *Journal of Human-Robot Interaction*, 2(2):103–128, 2013.

[24] Vijay Kumar, D. Rus, and Sanjiv Singh. Robot and sensor networks for first responders. *Pervasive Computing, IEEE*, 3(4):24–33, Oct 2004. ISSN 1536-1268. doi: 10.1109/MPRV.2004.17.

[25] Michael Lewis, Jijun Wang, and Paul Scerri. Teamwork coordination for realistically complex multi robot systems. In *NATO Symposium on Human Factors of Uninhabited Military Vehicles as Force Multipliers*, pages 1–12, 2006.

[26] Paul Maxwell, Anthony A Maciejewski, Howard Jay Siegel, Jerry Potter, and James E Smith. A mathematical model of robust military village searches for decision making purposes. In *IKE*, pages 311–316, 2009.

[27] Mehran Mesbahi and Magnus Egerstedt. *Graph theoretic methods in multiagent networks*. Princeton University Press, 2010.

[28] R. R. Murphy and J. L. Burke. The safe human-robot ratio. In M. Barnes and F. Jentsch, editors, *Human-Robot Interaction in Future Military Operations*, chapter 3, pages 31–49. Ashgate Publishing, 2010.

[29] Robin R Murphy. Rescue robotics for homeland security. *Communications of the ACM*, 47(3):66–68, 2004.

[30] Andrew L Nevai, Kevin M Passino, and Parthasarathy Srinivasan. Stability of choice in the honey bee nest-site selection process. *Journal of theoretical biology*, 263(1):93–107, 2010.

[31] S. Nunnally, P. Walker, M. Lewis, N. Chakraborty, and K. Sycara. Using Haptic Feedback in Human Robotic Swarms Interaction. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 57(1):1047–1051, September 2013. ISSN 1541-9312. doi: 10.1177/1541931213571233.

[32] Steven Nunnally, P Walker, Andreas Kolling, Nilanjan Chakraborty, Michael Lewis, K Sycara, and M Goodrich. Human influence of robotic swarms with bandwidth and localization issues. In *Proceedings of the 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*.

[33] Steven Nunnally, Phillip Walker, Nilanjan Chakraborty, Michael Lewis, and Katia Sycara. Using Coverage for Measuring the Effect of Haptic Feedback in Human Robotic Swarm Interaction. *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 516–521, October 2013. doi: 10.1109/SMC.2013.94.

[34] Wei Ren, Randal W Beard, and Ella M Atkins. A survey of consensus problems in multi-agent coordination. In *Proceedings of the 2005 American Control Conference*, pages 1859–1864. IEEE, 2005.

[35] C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *ACM Siggraph Computer Graphics*, volume 21, pages 25–34. ACM, 1987.

[36] Thomas D Seeley. *Honeybee democracy*. Princeton University Press, 2010.

[37] Tina Setter, Alex Fouraker, Hiroaki Kawashima, and Magnus Egerstedt. Haptic Interactions With Multi-Robot Swarms Using Manipulability. *Journal of Human-Robot Interaction*, 4(1):60, 2015. ISSN 2163-0364. doi: 10.5898/JHRI.4.1.Setter. URL `http://humanrobotinteraction.org/journal/index.php/HRI/article/view/186`.

[38] William M Spears and Diana F Spears. *Physicomimetics: Physics-based swarm intelligence*. Springer Science & Business Media, 2012.

[39] D. J. T. Sumpter. *Collective Animal Behavior*. Princeton University Press, Princeton, N.J., 2012.

[40] J. M. Whetten, M. A. Goodrich, and Y. Guo. Beyond robot fan-out: Towards multi-operator supervisory control. In *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, Istanbul, 2010.

[41] Alan FT Winfield and Julien Nembrini. Safety in numbers: fault-tolerance in robot swarms. *International Journal of Modelling, Identification and Control*, 1(1):30–37, 2006.

[42] Holly A Yanco and Jill L Drury. Classifying human-robot interaction: an updated taxonomy. In *SMC (3)*, pages 2841–2846, 2004.