Scale-Invariant Specifications for Human–Swarm Systems

JOEL MEYER [1], **AHALYA PRABHAKAR** [2], **ALLISON PINOSKY** [1], **IAN ABRAHAM** [2],
ANNALISA TAYLOR [1], **MILLICENT SCHLAFLY** [3], **KATARINA POPOVIC** [1], **GIOVANI DINIZ** [4],
BRENDAN TEICH [5], **BORISLAVA SIMIDCHIEVA** [5], **SHANE CLARK** [6],
AND TODD MURPHEY [1] (Senior Member, IEEE)

[1] Center for Robotics and Biosystems, Northwestern University, Evanston, IL 60208 USA
[2] Department of Mechanical Engineering and Materials Science, Yale University, New Haven, CT 06511 USA
[3] Department of Mechanical and Process Engineering, ETH Zurich, 8092 Zurich, Switzerland
[4] TurbineOne, San Francisco, CA 94105 USA
[5] Raytheon BBN Technologies, Cambridge, MA 02138 USA
[6] Skydio, San Mateo, CA 94402 USA
CORRESPONDING AUTHOR: JOEL MEYER (joelmeyer@u.northwestern.edu)

(Regular Article)

ABSTRACT We present a method for controlling a swarm using its spectral decomposition—that is, by describing the set of trajectories of a swarm in terms of a spatial distribution throughout the operational domain—guaranteeing scale invariance with respect to the number of agents both for computation and the operator tasked with controlling the swarm. We use ergodic control, decentralized across the network, for implementation. In the DARPA OFFSET program field setting, we test this interface design for the operator using the swarm tactics and offensive mission planning (STOMP) interface—the same interface used by Raytheon BBN throughout the duration of the OFFSET program. In these tests, we demonstrate that our approach is scale-invariant—the user specification does not depend on the number of agents; it is persistent—the specification remains active until the user specifies a new command; and it is in real time—the user can interact with and interrupt the swarm at any time. Moreover, we show that the spectral/ergodic specification of swarm behavior degrades gracefully as the number of agents goes down, enabling the operator to maintain the same approach as agents become disabled or are added to the network. We demonstrate the scale-invariance and dynamic response of our system in a field-relevant simulator on a variety of tactical scenarios with up to 50 agents. We also demonstrate the dynamic response of our system in the field with a smaller team of agents. Finally, we make the code for our system available.

INDEX TERMS Human–swarm interfaces, human–swarm systems, multirobot systems, swarm robotics.

I. INTRODUCTION

ONE of the biggest problems in the field of swarm robotics is that it is difficult for human users to reap the potential benefits of swarm robotic systems (i.e., robustness and information gains) due to the high cognitive load associated with controlling swarms [1]. Cognitive load has been shown to scale with both the size of the swarm that a user controls and the complexity of the environment in which the swarm and user operate (due to both agent–agent interactions and the length of time that the operator has to reason about decisions) [2]. Swarm operators need interfaces and control algorithms that are *scale-invariant* to enable operators to specify the same objective to swarms of different sizes without modification. Two major components are needed to create such a system—an interface to transform user input into scale-invariant commands and a control algorithm to

automatically and flexibly adapt user commands to changes in swarm size (which may occur due to communication dropouts, hardware failures, or newly available robots joining the swarm).

A. SCALE-INVARIANT USER INTERFACES

Prior work on user interfaces for swarm systems includes interfaces that enabled operators to control swarms of various sizes through haptic devices [3], [4], touch-based tablet applications [5], [6], and whole-body spatial gestures [7], [8]. Other researchers have run studies to find sets of gestures users found natural for controlling a swarm [9], [10] or developed programming languages that made it easier for users to control a robot swarm through a computer terminal [11].

While these methods enabled a direct mapping between human commands and swarm motion, these commands were predefined, and the human operator could not create new commands on the fly while they were controlling their swarm. The human operators also had to maintain constant situational awareness while controlling their swarm, and there was little or no persistent, autonomous swarm behavior in response to changes in the environment or swarm size.

Prior implementations of swarm interfaces in field scenarios also limited users by only allowing them to specify behavior for their swarm through predefined commands or fixed actions from a "playbook" [12]. Other swarm interfaces from the field were primarily designed to display information—they enabled small adjustments to ongoing missions but limited real-time human operator intervention overall [13], [14].

As a swarm interacts with both the environment and other agents in the environment, the operator's interface should enable the operator to specify behavior that is both scale-invariant with respect to swarm size and *persistent*— which gives the operator the option of performing other tasks, while the swarm executes their initial command. The interface should also enable users to modify these scale-invariant and persistent commands, while they are in progress if the environment or user's task changes. The user swarm interface that we extend in this article, initially introduced in [15], enables users to specify commands to their swarm as target distributions that can be deployed regardless of swarm size. The user can modify these commands in real time and attend to other tasks after they specify a command since the swarm will persistently converge to the target given by the user. User target distributions are not predefined. Users can draw their own target distributions, which may correspond to another user's high-level conception of how to execute a task even if the two users' target distribution drawings are qualitatively different.

B. SCALE-INVARIANT CONTROL ALGORITHMS

Previous work in swarm control algorithms has used methods that prespecify motion behaviors for each robot in the swarm as part of a formation for accomplishing a task [16], [17], [18], [19], [20]. Other previous work designed specifically for swarm sensing and coverage has relied on potential fields [21], [22], [23] and/or Voronoi partitions [24], [25], [26] to plan swarm trajectories. While these methods have also been used for applications such as target tracking [27] and swarm agent rendezvous [28], they were not designed for dynamic replanning with a human user or for replanning needs due to changes in swarm size from attrition or newly available agents joining the swarm. More recent swarm control methods have exhibited a greater degree of flexibility in adapting to changes in swarm size but are controlled via a central node, which leads to swarm system vulnerability [29], [30]. Other work on exploration and mapping [25], [31] incorporates decentralized methods that are robust to swarm agent communication dropouts and hardware failure. However, these methods do not enable a user to enter the swarm control loop and flexibly respecify swarm behavior as the swarm's environment and/or user objectives change.

Various attempts have been made to enable human users to flexibly respecify swarm behavior. In [32], a user can take control of one member of their swarm to influence or modify the task that the swarm as a whole is currently performing. This work bears similarities to other prior work involving leader–follower methods [33], [34] for controlling and influencing a swarm's behavior with individual swarm members or small subteams within the swarm. However, with an increasing number of agents, adjusting swarm behavior by influencing individual agents can become less effective if the user's swarm can lose these designated leader agents to attrition (due to communication dropouts, hardware failure, and so on) or if the user has to manually select these leader agents from their swarm in real time.

In contrast, defining how the collective swarm behaves using a spatial distribution has been shown to both scale favorably to swarms of different sizes and enable human users to dynamically replan with swarms. Spatial distributions (both static and dynamic) have been used to control swarms in sensing and coverage applications [35], [36], [37]. Dynamic distributions created by a human user have especially shown promise for real-time human swarm control. Diaz-Mercado et al. [38] present a decentralized, density-based coverage approach in which an operator uses a tablet interface to define an area for their swarm to explore. The user draws the desired exploration areas on the tablet interface, which are then partitioned into different regions using Voronoi tessellations. These tessellations are assigned to individual agents in the swarm for coverage. Each agent then needs to communicate with its nearest neighbors for the swarm to converge. This system as a whole was robust to agent dropouts due to decentralization, was demonstrated with swarms of different sizes, and enabled users to dynamically replan by redrawing the desired swarm coverage distribution on their tablet.

The method that we extend from [15] uses a decentralized ergodic coverage algorithm (first introduced in [39]), which contains many of the desirable attributes from [38] but does not need to partition the operating space and assign partitions

to each agent in the swarm. A robot running decentralized ergodic coverage spends an amount of time in different regions of the exploration space that is proportional to the spatial statistics measure of these regions in the exploration space, taking into account the consensus estimate of how much time the entire swarm has spent there. This implies that a human operator can use spatial distributions to specify how a swarm should allocate its agents to achieve proportional coverage of an exploration space in a decentralized manner. Unlike the algorithm presented in [38], swarms of agents using decentralized ergodic coverage do not need to communicate with their nearest neighbors to converge to a user's distribution—communication with any agents in the swarm will improve convergence [39]. Not using Voronoi partitions also enables individual agents in the swarm to explore the full environment without being bound to a particular assigned area. This is especially helpful under agent communication dropout or hardware failure since there is no need to recalculate or reassign partitions—our system will persistently and flexibly adapt to changes in swarm size and the user target distribution.

C. MAIN CONTRIBUTIONS

Our work develops a method for scale-invariant swarm control with a touch interface and a decentralized control algorithm. The touch interface enables users to prescribe strategies at the swarm level rather than at the individual agent level [15]. Gestures from the touch interface are converted to commands for the swarm using a decentralized ergodic coverage approach, which is invariant to swarm size [39]. Each agent responds to the desired coverage map in real time. We extend our prior work described in [15] by demonstrating the capability of our system with larger swarms with field-relevant executions in the swarm tactics and offensive mission planning (STOMP) command software.

Our main contributions are given as follows: 1) user interfaces for creating scale-invariant specifications for human–swarm systems and 2) a scale-invariant (with respect to the number of agents) integrated system architecture for real-time human swarm control that enables dynamic replanning with swarms. These two components enable human operators to dynamically replan with a swarm for exploration and distributed sensing by using ergodicity as a quantitative measure of information in the environment. Using our system, the operator maps visual input from both their line of sight in the environment and the command interfaces at their disposal (described in Section II-B) to a spatial representation of information that they then send as a command to their swarm via their user interface. Each member of the swarm runs a decentralized ergodic coverage algorithm (described in Section III) that transforms user commands into swarm trajectories. The combination of visual input that the operator receives, the user interfaces they send swarm commands through, and the decentralized ergodic coverage algorithm running on each member of the swarm enables the swarm to converge to the operator's spatial representation of

information regardless of how many agents in the swarm the operator has at their disposal at any point in time. Since the internal process that an operator uses to map visual input to expected information content is qualitative (in the sense that we do not have a model of how humans make these choices), our system affords flexibility to the operators—different operators can send different commands (representing spatial information) to their swarms based upon the same visual input.

II. SYSTEM OVERVIEW

In this section, we introduce the key elements of our system. First, we describe the user interfaces. Then, we describe the mission planning interface. Then, we discuss the individual swarm agents (rovers). Finally, we discuss the overall system architecture linking these components.
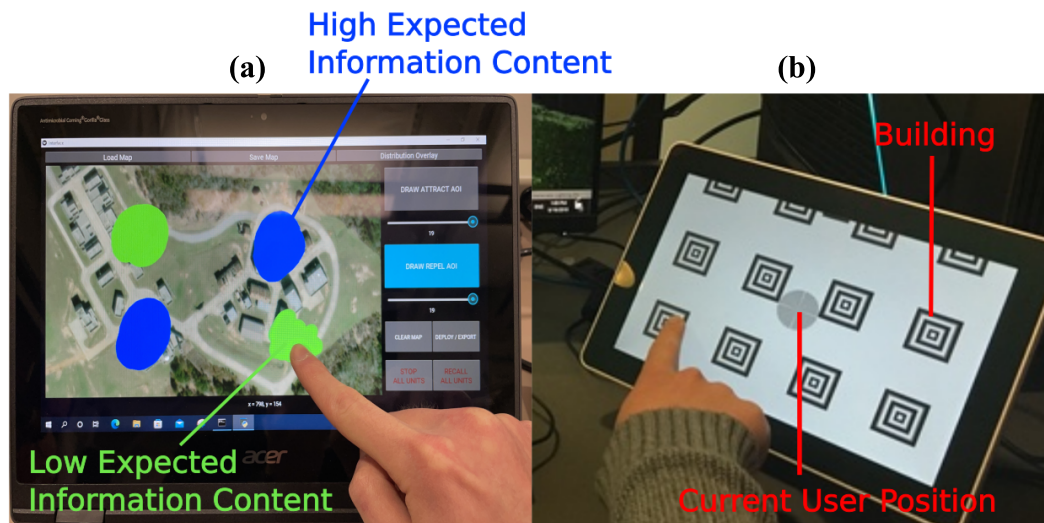
A. USER INTERFACE

Over the course of the DARPA OFFSET program, we developed two user interfaces. A touchscreen interface was used for experiments run with Raytheon BBN's STOMP command software—where the only difference between simulation and physical experiments is that STOMP is communicating with simulated agents. In addition, a Tanvas interface [40] with haptic feedback was used for physical hardware experiments at OFFSET field exercises. Both interfaces are shown in Fig. 1. Details of both interfaces are included in the following sections.

1) TOUCHSCREEN

The touchscreen interface shown in Fig. 1(a) transmits user specifications to the user's swarm of agents. "User specifications" (also known as "target distributions") are distributions in physical space that conveys areas the user wants their swarm to converge to or avoid for the purpose of exploration. The operator is able to draw these areas on the touchscreen with their fingertip, stylus, or computer mouse. Blue regions indicate exploratory areas (high expected information content) for agents, and green regions specify restricted areas (low expected information content) that the agents must avoid. "Expected Information Content" in this work is defined as the relative importance that an operator assigns to a particular area of the environment (via prior knowledge or intuition) for the purpose of exploration with their swarm. Areas with high expected information content may be expected to contain a high-value target, while areas with low expected information content may be expected to not contain a high-value target (or may be suspected to contain dangers that could disable the user's swarm).

The user can adjust the thickness of the lines that they draw while creating these regions (for finer or coarser specifications) by using the slider interface on the right-hand side of the touchscreen. User specifications that contain more than one target area (i.e., both a blue and a green region, multiple blue regions, multiple green regions, or a combination of the aforementioned) are "multimodal" and represent a

FIGURE 1. User specifications using touch interfaces. (a) Touchscreen interface, through which users can draw areas containing high (blue) and low (green) expected information content with their fingertips. Users draw over a map of the swarm's operating environment and can adjust their drawings in real time. (b) Tanvas interface, which renders information in the environment as changes in surface friction that the user feels as they slide their finger across the interface, which enables the user to localize themselves in the environment and issue commands without looking at the interface. The circle on the Tanvas interface represents the user's current position in the environment, while the squares represent buildings in the environment. Users can also draw high and low expected information content areas with the Tanvas. We used the touchscreen for our field relevant simulations in Section V and the Tanvas for field tests in Section VI.

multimodal distribution of expected information in physical space. Once the user finishes drawing these regions, they can tap/click the "Deploy" button, which sends the user's multimodal drawing (in 2-D array form) to each agent in the swarm through an ROS websocket. In addition, the touchscreen interface allows the user to generate an RGB image of the map with the user's multimodal distribution overlaid on top of it. This RGB image of the map with the user's multimodal distribution can be used for debugging purposes post-experiment or for situational awareness during testing to see how the user's drawing is translated into a multimodal distribution. User specifications are modifiable in real time; users can draw over regions that have already been drawn on even if the agents have not yet made it to those regions. Users can also clear the interface screen if they wish to redraw a particular target specification or create a new one.

The touchscreen interface application can be loaded on either tablets or PC's. Users can launch multiple instances of the interface application if the user wants to specify target distributions for different subgroups of the swarm (keeping two or more specification channels for these subteams open at the same time). The ability to use the touchscreen interface application on either tablets or PCs gives the operator flexibility in not being restricted to a particular operating system. A downside of the touchscreen interface is that there is no haptic feedback. The operator needs to continuously look at the screen, while they specify target distributions, meaning that the operator may struggle to respecify swarm behavior if their operating environment requires visual awareness.

2) TANVAS

The Tanvas [as shown in Fig. 1(b)], on the other hand, gives users haptic feedback reflecting real-time changes in the environment as they send commands to their swarm. The Tanvas renders textures on its smooth screen by modulating the friction underneath the user's fingertip [40]. These haptic features represent different objects in the environment, such as buildings, with different textures. The circle on the Tanvas interface screen represents the user's current position in the environment, whereas squares on the Tanvas interface screen represent buildings and other structures. The Tanvas renders all features on its screen with respect to the user's body frame. With the Tanvas, the user can specify regions of exploratory interest by double-tapping the screen, shading the region on the Tanvas, and double-tapping the screen again. At the second double tap, the user's specification coordinates are transformed into a distribution and then sent over a TCP socket to each agent in their swarm. The Tanvas also provides audio feedback to the user when they double-tap the screen.

Haptic feedback is critical for "boots on the ground." Operators need to be aware of their immediate surroundings while working and cannot afford to keep their eyes focused on an interface. The Tanvas enables users to orient themselves in the environment and send commands while watching what is going on around them without looking away, which, in the OFFSET scenario, was particularly useful for scenarios that required quick respecification of the swarm and that also required high-operator awareness of their surroundings due to drones flying close by. A downside of the Tanvas is that it represents the operating environment at a lower level of
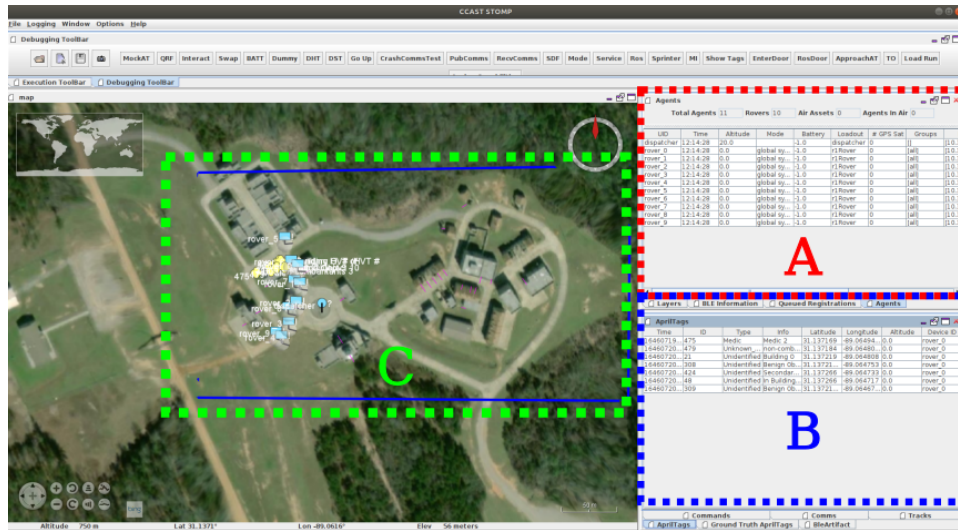
FIGURE 2. STOMP—a field relevant simulator for OFFSET. This figure shows an overview of the STOMP simulator. Users monitor the positions of their swarm and receive real-time information about the status of each agent, communications between agents, and information detected by agents as they operate in the environment. Boxes A and B show figures that display agent status, communications, and April Tag detections. Box C shows the agents' real-time positions in the operating environment.

granularity with haptic feedback than the touchscreen interface can with visual feedback.

B. STOMP

As a part of OFFSET, we fully integrated our decentralized ergodic coverage algorithm with Raytheon BBN's mission planning interface STOMP. Fig. 2 shows an overview of the STOMP environment. Box A shows the status of each rover, which includes its IP address and battery level. Box B displays information about different objects detected by the rovers. Since the objects are identified by April Tags, box B contains the ID number of each tag, the type of information the tag represents (medic, civilian, building, and so on), the global pose of the tag in the environment, and the ID number of the rover that detected it. Box C shows the current state of the operating environment, which includes real-time rover positions and detected April Tags.

C. ROVER

Fig. 3 shows the Aion Robotics R1 Rover that was used for field tests performed at DARPA OFFSET. The rover includes an NVIDIA Jetson TX-2 embedded computing device and a Pixhawk 2.1 controller with a HERE GPS unit running ArduRover 3.2. Onboard sensors include an RPLidar A1M8 2-D 360 LiDAR and an Intel RealSense D435i depth camera. Motion planning on the rover used the decentralized ergodic coverage algorithm, described in Section III, whose output was sent to a real-time rapidly exploring random tree (RT-RRT*) algorithm [41], described in Section III-C, for real-time path planning and obstacle avoidance.

The ergodic coverage algorithm ran locally on the NVIDIA TX-2 of each individual rover, which was each running a local ROS network. The rovers in the swarm communicated

with each other and received swarm-level commands from the human user over a local LTE network through a Java/Protelis interface developed by the Raytheon BBN team. The rovers connected to the LTE base station via custom-built USB LTE modems. The OFFSET experiments were conducted on a small range area that contained a combination of grassy terrain and concrete sidewalks. The rovers operated in a bounding box defined by GPS coordinates provided by Raytheon BBN's Java/Protelis interface. User commands were given at the Tanvas interface, which was connected to the robotic agents through the Java/Protelis interface and a TCP protocol.

D. UNIFYING SYSTEM ARCHITECTURE

Fig. 3 shows a unifying system architecture that contains the major components of our system, which were used throughout the OFFSET program. It also shows the entire sequence of human–swarm interaction for a user commanding a swarm with our system. Users start by inputting commands for their swarm through gestures at their user interface (either a Tanvas or touchscreen), which are transformed into target distributions representing swarm-level behavior and then sent to a communication node running on each rover. The algorithm running on each rover (explained in Section III) uses the target distribution that it has received via its communication node to determine its next control actions. Rovers send their past trajectories to each other (described in Section III), which enables the whole team of rovers to converge to the user's commanded target distribution. The user receives visual feedback on the progress of their swarm as it converges to their target distribution via the locations of the agents in their swarm through STOMP and the user's line of sight in the actual environment. The user also receives feedback on the
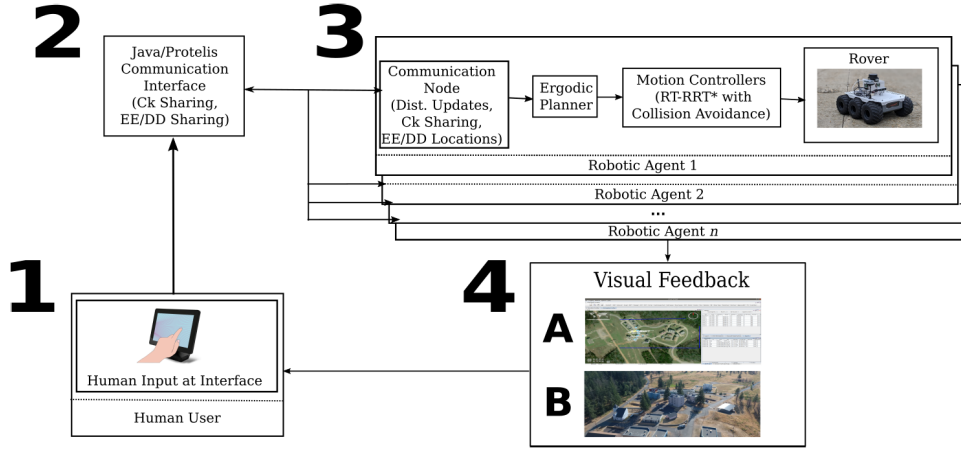
FIGURE 3. Unifying system architecture. This figure shows the entire sequence of human–swarm interaction in our system. In 1), the user starts by inputting their target distribution through an interface (either a Tanvas or touchscreen). The target distribution is sent to a communication node running on each rover, as shown in 2). The input is then transformed into a set of control commands via the algorithm running on each rover (explained in Section III) in 3). The operator uses visual feedback on the swarm's progress in 4)—provided through plots (introduced in Section V), the locations of the agents in STOMP 4A), and the operator's line of sight in the actual environment 4B)—to plan subsequent target distributions to send to the rovers to accomplish their desired task.

progress of their swarm via real-time data plots (introduced in Section V). All of this information will help the user determine the content and timing of their next commands/target distribution specifications as they complete their desired task.

III. ALGORITHM OVERVIEW

This section describes a decentralized ergodic coverage algorithm (first derived in [39]) that converts spatially distributed task information into temporally driven motion for a networked set of robotic agents. To integrate this algorithm with our system, we have extended the algorithm by augmenting its outputted agent trajectories with RT-RRT* for low-level obstacle avoidance. The inputs to the algorithm are high-level user task specifications that are distributions in physical space representing expected information content in the environment. Incorporating user task specifications into the decentralized ergodic coverage algorithm enables dynamic replanning with the user's swarm for exploration tasks.

A. ERGODICITY AND THE ERGODIC METRIC

We start by defining ergodicity and the ergodic metric. Assume that the state of a single robotic agent at time t is given by $x(t) : \mathbb{R}^+ \to \mathbb{R}^n$ and the controls given to the robot at time t are defined as $u(t) : \mathbb{R}^+ \to \mathbb{R}^m$. The dynamics of the robot are defined to be a control-affine dynamical system of the form

$$\dot{x}(t) = f(x(t), u(t)) = g(x(t)) + h(x(t)) u(t) \quad (1)$$

where $g(x) : \mathbb{R}^n \to \mathbb{R}^n$ is the free, unactuated dynamics of the robot and $h(x) : \mathbb{R}^n \to \mathbb{R}^{n \times m}$ is the dynamic control response subject to input $u(t)$. Define the robot's time-averaged statistics $c(s, x(t))$ for a trajectory $x(t)$ (i.e., the statistics describing where the robot spends most of its time)

for some time interval $t \in [t_i, t_i + T]$ as

$$c(s, x(t)) = \frac{1}{T} \int_{t_i}^{t_i+T} \delta(s - x_v(t)) \, dt \quad (2)$$

where δ is a Dirac delta function, $T \in \mathbb{R}^+$ is the time horizon, $t_i \in \mathbb{R}^+$ is the ith sampling time, and $x_v(t) \in \mathbb{R}^v$ is the robot's state that intersects with the exploration state space (with state dimension $v <$ less than dimension n of the exploration space). The ergodic metric introduced in [42] relates the time-averaged statistics $c(s, x(t))$ to an arbitrary spatial distribution $\phi(s)$ via

$$\mathcal{E}(x(t)) = q \sum_{k \in \mathbb{N}^v} \Lambda_k (c_k - \phi_k)^2$$
$$= q \sum_{k \in \mathbb{N}^v} \left(\frac{1}{T} \int_{t_i}^{t_i+T} F_k(x(t)) \, dt - \phi_k \right)^2 \quad (3)$$

where

$$\phi_k = \int_{\mathcal{X}_v} \phi(s) F_k(s) \, ds.$$

In this work, $\phi(s)$ is the operator's spatial representation of information in the environment based upon their line of sight in the environment and visual input that they receive from their command interfaces (see Section II-B)—in other works such as [43], $\phi(s)$ is constructed from an agent's sensor's measurement model. $q \in \mathbb{R}^+$ is a scalar weight on the metric, c_k is the Fourier decomposition of $c(s, x(t))$,[1] ϕ_k is the *spectral decomposition* of the spatial distribution $\phi(s)$, and

$$F_k(x) = \frac{1}{h_k} \prod_{i=1}^{v} \cos\left(\frac{k_i \pi x_i}{L_i}\right)$$

---

[1] The cosine basis function is used; however, any choice of basis function F_k that can be differentiated with respect to the state $x(t)$ and that can be evaluated along an agent's trajectory is acceptable.

is the cosine basis function for a given coefficient $k \in \mathbb{N}^\nu$. h_k is a normalization factor defined in [42], and $\Lambda_k = (1 + \|k\|^2)^{-(\nu+1/2)}$ are weights on the frequency coefficients. A robot whose trajectory $x(t)$ minimizes (3) as $t \to \infty$ is optimally ergodic with respect to the target distribution. That is, the robot spends an amount of time in different regions of the exploration space that is proportional to the spatial statistics measure of these regions in the exploration space.

Thus, we obtain a method for specifying how long a single agent should spend in different regions of the exploration space. If the user sends a multimodal target distribution to the system that conveys the relative "importance" of each area, minimizing the ergodic metric enables an agent to spend time in each area that is proportional to each area's expected information content. If the user needs to modify their target distribution, the agent will flexibly respond by changing its trajectory to match the new distribution of expected information content in the environment. Section III-B shows that in addition to allowing flexible task specification, the ergodic metric enables a user to control a decentralized network of agents.

B. DECENTRALIZED ERGODIC CONTROL

Consider a set of N agents with state $x(t) = [x_1(t)^\top, x_2(t)^\top, \ldots, x_N(t)^\top]^\top : \mathbb{R}^+ \to \mathbb{R}^{nN}$. The multiagent system's contribution to the time-averaged statistics c_k can be rewritten as

$$c_k = \frac{1}{N} \sum_{j=1}^{N} \frac{1}{T_\mathcal{E}} \int_{t_i}^{t_i+T} F_k(x_j(t)) \, dt$$

$$= \frac{1}{T_\mathcal{E}} \int_{t_i}^{t_i+T} \tilde{F}_k(x(t)) \, dt \tag{4}$$

where $\tilde{F}_k(x(t)) = (1/N) \sum_j F_k(x_j(t))$, N is the number of agents, T is the time horizon, t_i is the current time step, $\Delta t_\mathcal{E}$ determines how far into the past the agent needs to remember, and $T_\mathcal{E} = T + \Delta t_\mathcal{E}$. The c_k value used to calculate the ergodic metric is the average c_k value of all agents on the team. We use the spatial and temporal decompositions to calculate the ergodic metric for the multiagent team (3).

We determine the ergodic metric's time sensitivity by differentiating with respect to a fixed time duration λ and evaluating it at some fixed time t, which results in the costate equation. We can additionally show that each agent can generate an independent action that contributes to minimizing the entire team's ergodic metric. Let us first define the dynamics of the collective multiagent system as

$$\dot{x} = f(x, u) = g(x) + h(x) u$$

$$= \begin{bmatrix} g_1(x_1) \\ g_2(x_2) \\ \vdots \\ g_N(x_N) \end{bmatrix} + \begin{bmatrix} h_1(x_1) & \cdots & 0 \\ \vdots & \ddots & \\ 0 & & h_N(x_N) \end{bmatrix} u. \tag{5}$$

From work [39], [44], we can calculate the adjoint variable of the ergodic objective function as

$$\dot{\rho} = -2\frac{q}{T} \sum_{k \in \mathcal{N}_\nu} \Lambda_k (c_k - \phi_k) \frac{\partial \tilde{F}_k}{\partial x} - \frac{\partial \Phi}{\partial x} - \frac{\partial f}{\partial x}^\top \rho(t) \tag{6}$$

with $\rho(t_i + T) = 0 \in \mathbb{R}^n$.

Here, $f(x, u)$ represents the dynamics of the multiagent team and Φ is a barrier function that is needed to keep the agents within their operating environment (this barrier function is needed because of the spatial periodicity of Fourier transformations). Since the dynamics of each agent in the multiagent team are independent of one another, we define $(\partial \tilde{F}_k / \partial x)$ and $(\partial f / \partial x)$ as

$$\frac{\partial \tilde{F}_k}{\partial x} = \frac{1}{N} \begin{bmatrix} \dfrac{\partial F_k(x_1)}{\partial x_1} \\ \vdots \\ \dfrac{\partial F_k(x_N)}{\partial x_N} . \end{bmatrix} \text{ and } \frac{\partial f}{\partial x}$$

$$= \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & 0 & \cdots & 0 \\ 0 & \dfrac{\partial f_2}{\partial x_2} & & \\ \vdots & & \ddots & \\ 0 & & & \dfrac{\partial f_N}{\partial x_N} . \end{bmatrix} .$$

Using these definitions, we can solve for an expression for the state adjoint variable ρ_j for each agent

$$\dot{\rho}_j = -2\frac{q}{T} \sum_{k \in \mathcal{N}_\nu} \Lambda_k (c_k - \phi_k) \frac{\partial F_k}{\partial x_j} - \frac{\partial \Phi}{\partial x_j} - \frac{\partial f_j}{\partial x_j}^\top \rho_j(t). \tag{7}$$

As a result, following the work from [39], we can now define a controller for the collective swarm that minimizes the ergodic metric

$$\begin{bmatrix} u_{\star,1}(t) \\ \vdots \\ u_{\star,N}(t) \end{bmatrix} = -R^{-1} \begin{bmatrix} h_1(x_1) & \cdots & 0 \\ \vdots & \ddots & \\ 0 & & h_N(x_N) \end{bmatrix}^\top \begin{bmatrix} \rho_1(t) \\ \vdots \\ \rho_N(t) \end{bmatrix}$$

$$+ \begin{bmatrix} u_{\text{def},1}(t) \\ \vdots \\ u_{\text{def},N}(t) \end{bmatrix} \tag{8}$$

where $R \in \mathbb{R}^{mN \times mN}$ is a diagonal matrix representing the weights on each swarm member's control inputs, mN is the size of the swarm system control input, and $u_{\text{def},j}$ is some default control policy running on each agent.

Since $h(x)$ is block diagonal, (8) becomes

$$u_{\star,j}(t) = -R_j^{-1} h_j(x_j)^T \rho_j(t) + u_{\text{def},j}(t) \tag{9}$$

for each agent $j \in [1, \ldots, N]$ and $R_j \in \mathbb{R}^{m \times m}$. Note that the jth agent does not depend on the ith agent. All that is required is that we communicate the c_k values between each agent to

obtain the time-averaged statistics of the swarm as a whole, which are used by each agent to calculate its own control values. Calculating the time-averaged statistics of the swarm as a whole has worst case linear computational complexity $\mathcal{O}(N)$ with respect to the number of agents N since the c_k for the swarm is calculated by averaging over the c_k values from individual agents. However, since real-world agents have limitations on their communication range (and can drop off the communication network), the computational complexity will be sublinear in practice since each agent will be calculating an average c_k over less than the total number of agents in the swarm. Since each agent calculates both its own Fourier history and set of controls for minimizing the swarm's ergodic metric, the computational complexity of these components of the decentralized ergodic control algorithm will be constant time or $\mathcal{O}(1)$ with respect to the number of agents in the swarm. The computational complexity of these components will only scale with the dimension of an agent's state (which is not time-varying for these demonstrations) and the number of Fourier decompositions that are used for each dimension of the exploration space, which scales with order $\mathcal{O}(v^k)$.

If a single agent detaches from the communication network, then that agent will minimize its own ergodic metric. Communication from other agents in the swarm helps the individual agent minimize the ergodic metric of the collective as it moves through the exploration space since it knows where other agents have recently been (and where they plan to go) from their communicated c_k values. This enables the swarm as a whole to converge to the user's target distribution. Section III-C describes the local planner we used for obstacle avoidance and low-level control.

C. INTERFACING WITH RT-RRT*

The control signal $u(t)$ for each agent is converted to kinematic input $\dot{x}(t)$ where the forward simulation $x(t)$ for each individual agent is supplied to the RT-RRT* low-level planner as a set of target controls. All safety features, such as rover velocity limits and obstacle avoidance, are handled *after* the user target specification is transformed by the ergodic coverage algorithm. Thus, the controls that are sent to the rover may be different from the output of the ergodic coverage algorithm. This allows the ergodic coverage algorithm to avoid considering obstacle avoidance and divides the computational load into two segments: low-level planning for robot control and obstacle avoidance, and high-level task planning and adaptation with the ergodic coverage algorithm, which will persistently explore according to the spatial measure defined by the target distribution. Since the ergodic coverage algorithm is temporally driven (i.e., the amount of time spent in a region directly impacts future behavior), and because we replan at every timestep, any regions the user wants their swarm to explore, which cannot be immediately visited due to obstacles, will eventually be visited once the RT-RRT* planner calculates a path around the obstacles (assuming an unobstructed path from the agent's current position to these regions exists).

D. HIGH-LEVEL PLANNING FOR DYNAMIC-TASK ADAPTATION

This section defines how tasks are specified as spatial distributions represented by $\phi(x)$. We also define how new user commands representing new or updated tasks are combined with existing task specifications to enable multimodal distributions to act as descriptions of where agents need to be allocated. We focus on two scenarios: reallocating priority to "attraction" regions with high expected information content and lowering priority for "repulsion" regions with low expected information content.

1) SPECIFYING TASKS WITH EASTER EGGS AND DISABLING DEVICES

We use Easter eggs (EEs) to denote generic "attraction" regions and disabling devices (DDs) to denote generic "repulsion" regions, each of which are Gaussians centered at (latitude, longitude) coordinates in the environment. The positions of EEs and DDs that are specified by the user, or autonomously detected by an agent in the swarm, are sent to each agent in the swarm's network. The locations of these elements in the environment are added to the specification $\phi(x)$ by parameterizing the expected information content in the environment as a multimodal sum of Gaussians

$$\phi(x) = \frac{1}{\eta} \sum_i a_i \exp\left(-\frac{1}{2}\|x - x_{\text{EE}}\|^2_{\Sigma^{-1}}\right) \\ + \frac{1}{\eta} \sum_j b_j \left(1 - \exp\left(-\frac{1}{2}\|x - x_{\text{DD}}\|^2_{\Sigma^{-1}}\right)\right) \tag{10}$$

where η is a normalization factor, $\sum_i a_i = 1$ and $\sum_j b_j = 1$, and $x_{\text{EE}}, x_{\text{DD}}$ are the locations of EEs and DDs, respectively. The parameter Σ is the width of the Gaussian region of attraction or repulsion that can be tuned based on the size of the environment and the user's desired granularity. We used $\Sigma = \text{diag}(0.01, 0.01)$ for both EEs and DDs. EEs represent high expected information content regions (attraction regions), while DDs represent low expected information content regions (avoidance regions). The resulting distribution is then normalized and represented with ten Fourier coefficients for each dimension of the exploration space. Exploration space coordinates are transformed and scaled to a bounding box of size $[0, 1]^2$ for numerical stability.

2) SPECIFYING TASKS THROUGH USER DRAWINGS AT THE TABLET INTERFACES

The interfaces described in Section II-A transmit a user's specification (represented as a multimodal distribution drawn by the user on an interface screen) to the team of agents that the user is controlling. The user's multimodal task distribution consists of a discretized grid of normalized values that represent the relative importance of different areas of the environment (with respect to each other) that the team of agents operates in. When a user draws an "attraction"
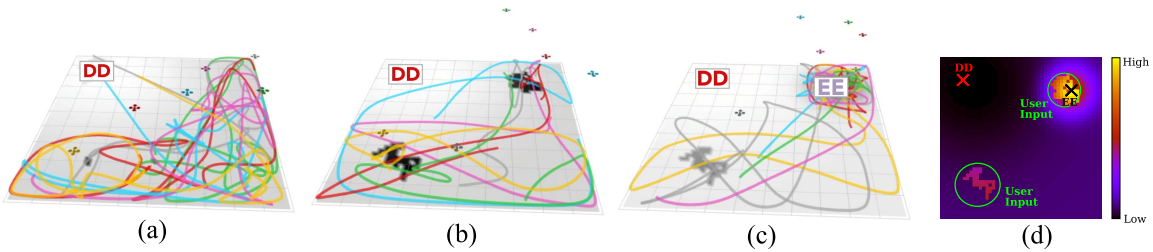
FIGURE 4. Simulation of a team of six agents dynamically adapting to the environment while also responding to user commands. (a)–(c) Time lapse of the agents' trajectories in response to the different stimuli. (a) Agents discover a DD and explore the rest of the environment while avoiding that location. (b) User inputs a bimodal target with two areas with high expected information content for the team of agents to explore. The team responds to the user command while continuing to avoid the DD location. (c) Team discovers an EE (in this example, the EE has higher expected information content than the user's attraction region drawing input—for other cases, EEs and user attraction regions have an equivalent expected information content) in the environment. The team of agents converges on the EE while still exploring the user drawing inputs to a lesser extent. The agents continue to avoid the DD location. (d) Target distribution containing the DD, user input, and EE. The "x" label in red marks the DD location, while the "x" label in black marks the EE location. The circles in green highlight the user target input from the Tanvas interface.

region on the interface screen, the areas of the interface that their finger makes contact with are assigned a value of 1 (the highest level of importance)—this is equivalent to the EE specifications described in the section above, the difference being that user attraction drawings do not have to be Gaussians centered at a particular position of the environment; drawings can take on different shapes that do not resemble Gaussians. Other areas that have not been drawn in are assigned a random noise value between [0, .001) in order for the team of agents to generate minimal coverage over the non-"attraction" areas. When a user draws a "repulsion" region, the areas of the screen that their finger makes contact with are assigned a value of 0—which is equivalent to the DD specifications described above (with other areas also being assigned random noise between [0, .001)). If both "attraction" and "repulsion" regions are drawn, areas that have not been drawn on are assigned a value of 0.5, which denotes "medium" information content. As above, the resulting distribution of attraction and repulsion regions is normalized and represented using ten Fourier coefficients for each dimension of the exploration space.

IV. SMALL-SCALE SIMULATIONS IN ROS

We demonstrate the dynamic response of our system with small-scale proof-of-concept tests in ROS simulations with six agents. We use aerial vehicles for these small-scale tests (as opposed to the ground vehicles used for our simulated demonstrations in Section V and field demonstrations in Section VI) because the dynamic response of our system is clearer with faster platforms that do not have to avoid obstacles in the environment. These agents explore an environment and adapt their trajectories to environmental stimuli and user commands sent from a Tanvas interface. Fig. 4(a) shows the team of six agents discover a DD (which we use to represent an area of the environment with the lowest possible expected information content) in the upper left corner of the environment.
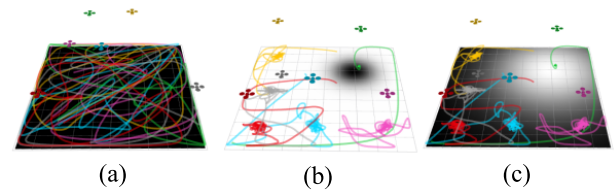
FIGURE 5. Heterogeneous team discovering a DD then dynamically responding to it based on team members' capabilities. The figures here show a time lapse of the agent trajectories. The team contains a DD blocker agent (agent 1) and non-DD blocking agents (agents 2–6). Agents 1–6 will respond differently to DDs. The team starts by uniformly exploring the environment in (a). Here, the black colored background corresponds to uniform information content in the environment for all of the agents, which leads to uniform coverage. (b) Agents discover a DD in the upper right-hand corner of the environment. The background here shows the information content in the environment from the perspective of the DD blocker agent—the white color corresponds to low information content, while the black color corresponds to high information content (the position of the DD). In this example, information content in the environment denotes where agents in the swarm should position themselves according to their capabilities. The DD blocker agent converges on the DD location (denoted by the black circle), while the other agents scatter and explore areas away from the DD. (c) Information content of the environment from the perspective of the non-DD blocking agents. The non-DD blocking agents avoid the white-colored area of the environment denoting low information content (where the DD is) and explore the black areas of the environment away from the DD. (a) Uniform exploration: agents 1–6. (b) DD convergence: agent 1. (c) DD avoidance: agent 2–6.

The agents avoid the DD and explore the rest of the environment. The user then sends a bimodal target distribution to the team of agents, represented by the black-shaded areas. In Fig. 4(b), the agents converge to the user's target distribution, while they continue to avoid the DD. In Fig. 4(c), the agents discover an EE—which we use to represent an area with higher expected information content than the targets sent by the user (the difference in expected information content
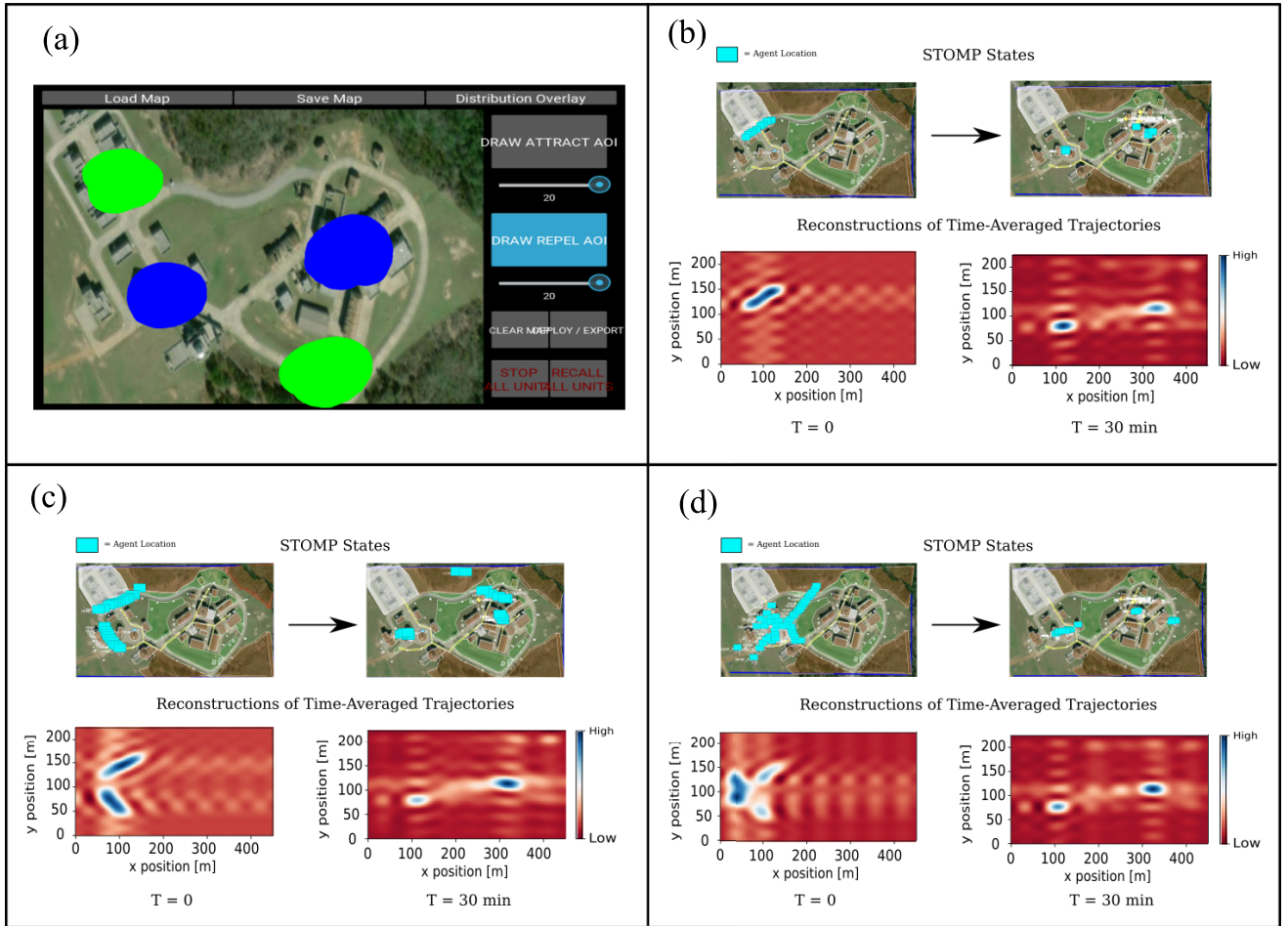
FIGURE 6. Scale invariance in target distribution convergence with respect to swarm size. Swarms of different sizes will all converge to the user's target distributions. The user specifies an example target distribution [shown in (a)], which contains two areas with high expected information content (shown in blue) and two areas with low expected information content (shown in green). (b)–(d) Swarms containing 10, 25, and 50 agents converging to this user target distribution. The top row of each of these (b)–(d) shows the initial positions of agents in the swarm (denoted by the blue rectangles) at time 0 s and their final positions after converging to the user's target distribution at 1800 s. The bottom row of each of the figures shows the Fourier reconstruction of the time average of the agents' trajectories in these swarms at 0 s (on the bottom left) and at 1800 s (at the bottom right). Despite the different sizes, all three swarms converge to the user target, as seen by the near identical Fourier reconstructions of the swarms' time-averaged trajectories at 1800 s.

between EEs and user attraction region drawing inputs is only different for this particular experiment—in all other cases, EEs and user drawings that represent attraction regions have the same expected information content values). The team of agents converges on this EE but continues to explore near the previous targets sent by the user. The team of agents spends more time near the EE due to its higher expected information content. The team of agents also continues to avoid the DD. Fig. 4(d) shows the distribution representing the agents' expected information content for the entire environment.

We then simulate a team of six heterogeneous agents exploring and adapting to environmental stimuli based on their individual capabilities. The team of six agents contains five regular agents tasked to explore the environment, while one agent is tasked to remove DDs from the environment. We simulate an agent discovering a DD and communicating its location to its teammates. The regular agents avoid the DD location, while the DD blocker agent converges on it to make it safe for any agent that comes close to it.

In Fig. 5(a), the team of heterogeneous agents uniformly explores the environment. When a DD location is discovered [see Fig. 5(b)], the DD blocker agent (the green-colored agent) converges on the DD's location to remove it from the environment, while the rest of the team continues to explore away from the DD [in Fig. 5(c)]. The examples shown in Figs. 4 and 5 demonstrate our system responding to detected changes in the information content of the environment with homogeneous and heterogeneous teams of six agents. Section V describes how we use STOMP to interface with a larger number of agents and demonstrate the scale-invariance of our system.

V. DEMONSTRATIONS USING STOMP
In this section, we demonstrate scenarios that are relevant to the OFFSET program by using STOMP to interface with up to 50 simulated agents. We demonstrate the high-level benefits that our system offers in scenarios that include covering multiple areas of the environment, converging upon a
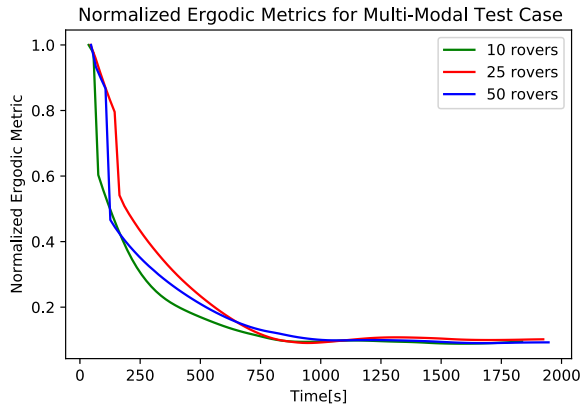
FIGURE 7. Normalized ergodic metric values converge across the scale. Swarms containing 10, 25, and 50 agents converged to the operator's desired multimodal target distribution (given in Fig. 6), as seen through the normalized ergodic metric values over time for all three swarm sizes asymptotically converging to 0. Despite the small difference in the time at which the operator's multimodal target distribution was sent to each differently sized swarm, all three swarms converged to the operator's target after approximately 1200 s (the experiments for each swarm size lasted for roughly 1800 s in total).

target area and quickly evacuating from it, and simultaneously converging on and surrounding a target area, one of the stated goals for the FX-5 exercises.[2] We also show how information collected by the swarm varies according to the size of the swarm and how our system automatically adapts to an operator's target specification if the swarm loses agents.

A. USER SPECIFICATIONS ARE SCALE-INVARIANT WITH RESPECT TO SWARM SIZE

In swarm exploration scenarios, the operator may want to specify which areas of the environment are more or less likely to contain information. Our system enables an operator to draw an arbitrary, multimodal distribution that contains areas with both high and low expected information contents for their swarm to explore. Our decentralized coverage algorithm [39] plans trajectories for the swarm such that the time agents spend in different areas of the environment is proportional to their expected information content. In the OFFSET setting, this is relevant for operators who seek to use swarms of robots to discover where targets of interest are and which areas should be avoided due to threats. We demonstrate an operator sending a multimodal target distribution to their swarm that contains two areas with high expected information content (which the swarm will spend more time in) and two areas with lower expected information content (which the swarm will spend less time in) on swarms that contain 10, 25, and 50 agents. The results are shown in Fig. 6.

The multimodal target shown in Fig. 6 contains two regions with high expected information content (blue) centered on clusters of buildings in the STOMP environment and two

[2]We did not perform physical experiments in FX-5 due to the COVID-19 pandemic. The code for our system can be found at https://github.com/MurpheyLab/rover_decentralized_ergodic_control.
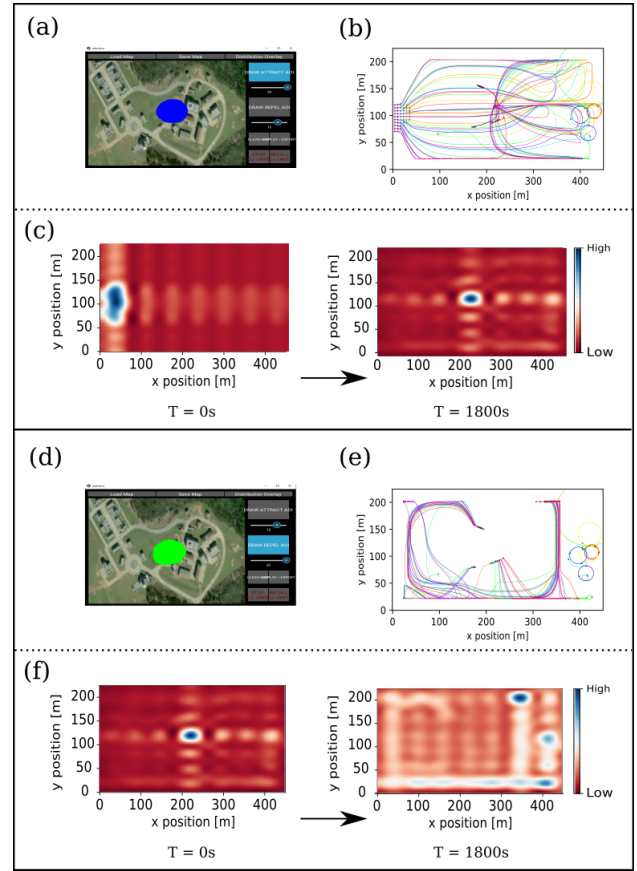
FIGURE 8. Dynamically responding to user target respecification. This figure shows a user specifying their swarm of 50 agents to converge upon a specific area in the center of the environment—the user then respecifies the center as an area to avoid, which causes their swarm to scatter. (a) Initial user target specification, which is a convergence region ("attraction" regions are drawn in blue in the interface). (b) Swarm agent trajectories responding to this target—the agents move from their initial positions on the left-hand side of the environment to the center (the different colored lines represent the trajectories of different agents). (c) Fourier reconstruction of the time average of these agent trajectories at time $t = 0$ s and $t = 1800$ s. (d) User respecifies their target to be an avoidance region ("repulsion" regions are drawn in green), which is located at the area that they previously specified to be a convergence region. (e) Agent trajectories moving away from the center of the environment in response to this new target. (f) Fourier reconstruction of the time-averaged agent trajectories for this new repulsion target. A video of this sequence can be viewed online **https://sites.google.com/view/scale-invariant-human-swarm.**

areas with low expected information content (green) centered on featureless areas in the environment. The operator monitors the swarm's "progress" with respect to the user's target specification through both the normalized ergodic metric plot (which indicates convergence when the steady-state normalized ergodic metric value approaches 0 or plateaus after decreasing for some time) and the plot showing the Fourier reconstruction of the time average of the agents' trajectories (which provides visual feedback on how closely the Fourier reconstruction of the swarm's time-averaged
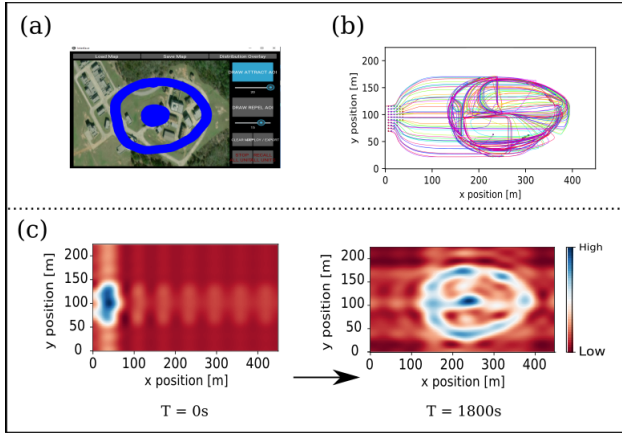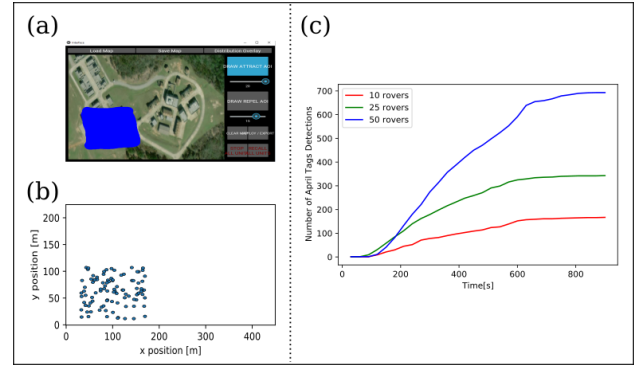
FIGURE 9. Nontrivial multimodal demonstrations. Our system enables swarms to converge to "nontrivial" multimodal target distributions. (a) User's target distribution, which resembles a bullseye. (b) Agent trajectories as the swarm converges to this target (the different colored plot lines represent the trajectories of different agents). (c) Fourier reconstruction of the time-averaged trajectories of the swarm at $t = 0$ s and $t = 1800$ s (the time at which the system has converged). As seen in (c), our system automatically allocates a small number of agents to converge on the smaller centroid of the bullseye and a larger number of agents to converge on the outer ring that surrounds the centroid. (b) Agents in the smaller centroid and outer ring exchange positions with one other as time tends toward infinity, which indicates that our system is persistent and will constantly work toward converging to the target.

FIGURE 10. Graceful degradation in the amount of information collected across swarms of different sizes. The user sends the same target distribution (a) to swarms of 10, 25, and 50 agents. The user expects there to be information (in the form of April Tags in this example) in the bottom left corner of the environment. (b) Ground-truth distribution of April Tags in the environment. As the swarms of different sizes converge to the user's target distributions, they detect the April Tags in the environment. (c) Number of April Tag detections for each swarm after 900 s. The swarm of 50 agents detects the most April Tags, followed by the swarms with 25 and 10 agents. The April Tag detection curves for all three different swarm sizes have the same shape and trend, which indicates that there is a graceful change in information detection across swarm sizes. Thus, an operator can expect larger swarms to detect more information but will not have to alter their strategy or mental model with smaller swarms.

trajectories matches the operator's target specification). The user knows that the swarm has converged to their target specification (and completed the current task) when the Fourier reconstruction plot of the time average of the agent trajectories in the swarm matches the target specification that the user has given through their interface or when the value of the normalized ergodic metric plot for their system plateaus remains relatively constant (after decreasing for some time).

Fig. 7 shows a plot of the normalized ergodic metric values over time for three different swarm sizes (10, 25, and 50 agents) running the same multimodal target distribution in the STOMP environment. As seen in Fig. 7, the normalized ergodic metric values of all three swarm sizes converge to near zero at roughly the same time. The Fourier reconstructions of the time average of the agents' trajectories (shown in the "Reconstructions of Time-Averaged Trajectories" plots in Fig. 6) of all three swarms also closely match the operator's original multimodal target specification at the end of each run. Large portions of the agent trajectories for each run are spent in areas of the map with high expected information content. The agent trajectories avoid areas with low expected information content. This indicates that an operator can expect the same system behavior from swarms of different sizes.

B. DYNAMICALLY RESPONDING TO USER TARGET RESPECIFICATION

In real-world operations, operators need to be able to quickly retask their swarm in response to changing environments.

We present an example scenario in which a swarm is given an area with high expected information content in the center of the map that then inverts and becomes an area with low expected information content (an area to avoid). This scenario is relevant to OFFSET in that a rover may discover a DD in an area that was previously thought to be "safe." The system would have to quickly respond to this new knowledge. It is difficult to achieve this type of behavior with geometric (Voronoi) methods [24], such as leader–follower methods for dispersion [45], or methods in which a user controls (or selectively takes control of) a single agent [32]. Fig. 8 demonstrates this behavior with our system for a team of 50 agents, showing the transition between the team converging on an attraction region and then scattering when the attraction region inverts into a repulsion region. This example is a larger scale version (in terms of number of agents and operating environment) of the example introduced in Section IV.

### C. NONTRIVIAL MULTIMODAL DISTRIBUTIONS (BULLSEYE)

An advantage of our system is the flexibility it provides operators—allowing operators to task swarms to explore areas of any arbitrary shape. Fig. 9 shows a swarm of 50 agents converging on a target that resembles a "bullseye" with two regions to converge to: a circle surrounded by an outer ring. This target distribution is analogous to a tactical situation in which a swarm must converge on a
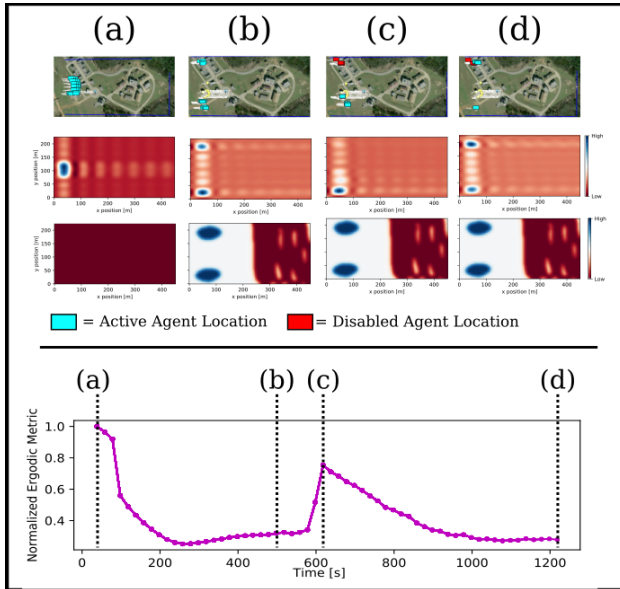
FIGURE 11. **Recovery from agent failure. In column (a) of the top half of this figure, the operator starts by specifying a multimodal distribution with two areas containing high expected information content in the top and bottom left corners of the environment. Their swarm of 25 agents converges to this target distribution, shown in column (b). In column (c), an EMP (electromagnetic pulse) device disables all of the agents in the top left corner of the environment. The swarm automatically adapts to the drop in the number of accessible agents by reallocating agents from the bottom left corner of the environment to the top right corner. The system then successfully converges to the original target, as seen in column (d). The bottom row of the figure shows the normalized ergodic metric plot for the swarm over the course of this experiment. The normalized ergodic metric starts at 1.0 at the timestamp shown in column (a). The normalized ergodic metric smoothly decreases to the value corresponding to column (b) as the system converges to the user target distribution. After the EMP occurs, there is a spike in the normalized ergodic metric plot [column (c)]. The system is now "far" from converging because all of the agents in one of the target areas have been disabled. However, the system is not as far as it was from converging at the start of the experiment in column (a) since some of the agents are still located at one of the user target areas. After the swarm reallocates members to the top left corner, the normalized ergodic metric converges to near zero again, as shown in column (d).**

target while surrounding it, perhaps to prevent the target from escaping [46]. As shown in Fig. 9, our system enables the swarm to successfully converge to this target. Also of note is that the agents will exchange positions with other agents between the center and outer ring of the bullseye as the system converges, which indicates that the swarm automatically adjusts to real-time changes in its time-averaged behavior. Thus, the operator does not need to manually reallocate members of the swarm, a task that could be mentally taxing as the total number of agents increases. Instead, the operator can focus on high-level strategy and interpreting the information that their swarm collects from the environment.
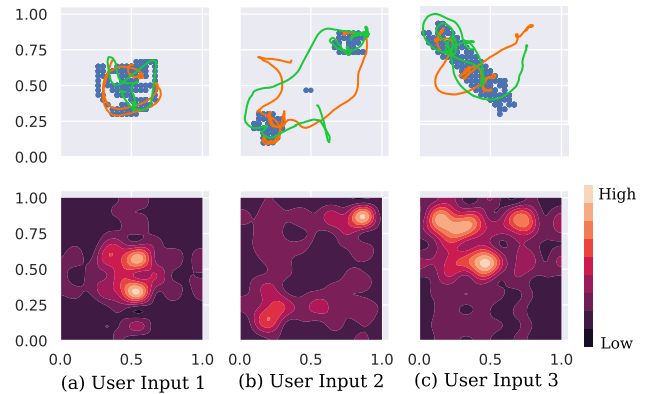

FIGURE 12. **Real-world agents dynamically respond to the user's target distribution sent through the Tanvas. (Top) Time lapse of the real-world agents' trajectories (shown in orange and green). (Bottom) Fourier reconstruction of the time average of these trajectories. When the user updates their target distribution (represented in (a)–(c) with the blue dots), the robots update their trajectories to converge to this new target.**

D. GRACEFUL DEGRADATION IN THE AMOUNT OF INFORMATION COLLECTED ACROSS SWARMS OF DIFFERENT SIZES

We also demonstrate the graceful degradation of our system—our system continuing to operate under unexpected agent loss and enabling the operator to use the same strategy or mental model with swarms of different sizes—with respect to the amount of information that the swarm collects. For "information," we use simulated April Tags that represent different entities relevant to the OFFSET program (IED, high-value target, and so on). We randomly distribute these April Tags throughout the STOMP simulation environment. As shown in Fig. 10, the operator draws a target distribution that uniformly covers an area of the map in which they expect their swarm to obtain information. As the figure shows, the largest swarm detects the highest number of April Tags. However, there was a graceful degradation in the amount of information collected for the smaller swarms, indicating that our system is suitable for uncovering information at any scale and that an operator can use our system with different numbers of agents, knowing that the amount of information collected will gracefully increase or decrease if new agents are added to the swarm or if agents are disabled during operation.

E. SCALE-INVARIANT CONTINUITY IN THE EVENT OF AGENT FAILURE

The scale-invariant continuity of our system also enables operators to retain the same strategy for executing a task even if agents in their swarm unexpectedly fail. Fig. 11 shows the results of an operator running a multimodal target distribution with 50 agents and then losing 25 of them to attrition (due to a simulated EMP) as they converge toward the target. The figure shows a spike in the normalized ergodic metric, which indicates that the swarm, now containing fewer agents,
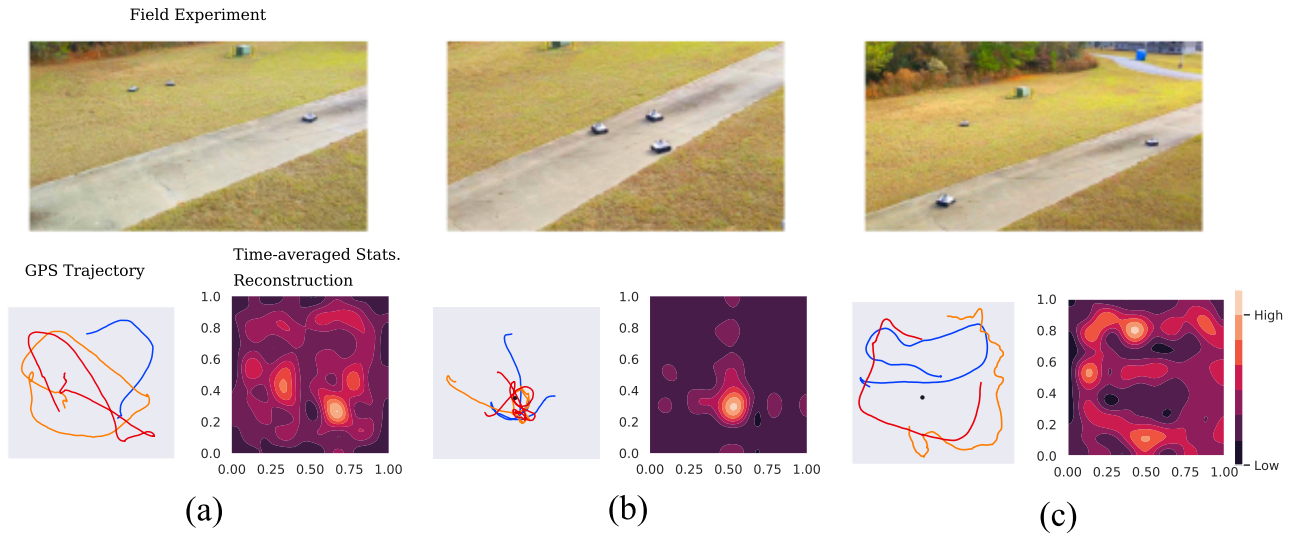
FIGURE 13. Inverting a user specification produces the same behavior in the real world as it does in simulation. The real-world rovers converge on a target with high expected information content. This target then transforms into a DD that the rovers must avoid. The top figure shows pictures of the real-world testing environment. In each bottom figure, the left shows the robots' trajectories (in red, orange, and blue), while the right shows the Fourier reconstructions of the time average of these trajectories. In (a), the real-world rovers start by uniformly covering their environment. We then add an area with high expected information content (which could represent a high-value target) via the command line to the center of the environment. The rovers update their target distribution with this high-value target location [shown in (b)] and converge toward the center. We then change the high-value target to a DD. (c) Rover trajectories and the Fourier reconstruction of the time average of these trajectories after 40 s—which was the approximate time that the rovers needed to move away from the location. (a) Exploration. (b) Discovery of EE. (c) EE becomes DD.

is further from converging to the target distribution than the original team was at the instant before 25 of the agents were disabled. Despite this, the system is able to continue converging toward the original target without user intervention, which indicates that our system can "recover" from the sudden loss of agents without methods such as network topology reconfiguration [47]. Thus, an operator does not have to abandon their current specification (or mental model of how to complete a task) if they lose agents. The remaining agents will converge to the existing target given enough time.

VI. FIELD DEMONSTRATION

We demonstrated our system at the FX-3 OFFSET field test in December 2019. The purpose of this demonstration was to enable operators to control the robots from the tablet interface (though we mention here that some of the testing included users sending commands to their robots via the PC command line by typing in a set of Gaussian means and covariances parameterizing the user's target distribution). We demonstrated two scenarios: one where the robot swarm converges on a set of dynamically changing high-value target areas and one in which the swarm converges on a single high-value target area and then scatters from the area once a threat is discovered.

In Fig. 12, several real-world rovers adapt to the operator's specifications, which are represented by the dotted blue areas on the top row of Fig. 12. As the real-world rovers receive new inputs through the Tanvas tablet interface, they update their target distributions and spend more time in areas of the environment with higher expected information content. For this real-world demonstration, we started with four real-world rovers, which then dropped down to two due to hardware issues. Despite the hardware issues, our system was able to adapt to the user target specifications with the drop in team size. The remaining agents updated their trajectories to compensate for the loss of their two team members.

Next, we demonstrate a team of four real-world rovers exploring their environment and dynamically adapting to environmental stimuli that they encounter in real time. Through the command line, we simulate one of the rovers discovering a high-value target and then sending the location of this high-value target to its teammates. Each member of the team updates its own target specification. We then transform the high-value target location into a DD, which causes the team members to scatter to avoid the area.

Fig. 13 shows the results at each stage of the scenario described above. The rovers start out uniformly exploring the task space [see Fig. 13(a)] until a high-value target is discovered [see Fig. 13(b)]. The rover that discovered the high-value target shares the target's location with its teammates. Each rover in the team updates its target distribution in a decentralized manner. Note that when the rovers converge to the target location, they do not move to the peak and wait—instead, they generate persistent, exploratory trajectories over the environment [see the bottom row that contains the Fourier reconstruction of the time average of the rover trajectories (see Fig. 13)]. We then transform the high-value target into a DD, which causes the rovers to scatter [see Fig. 13(c)].

As the rovers scatter, they uniformly explore areas of the environment that are away from the DD. These examples show that our system works in the real world, in real time, with a human operator in the loop.

VII. DISCUSSION, LESSONS LEARNED, AND FUTURE WORK

In this work, we developed a scale-invariant system for real-time human–swarm control. Our touchscreen interface enables users to specify behavior for a swarm in a setting that is scale-invariant—the user specification does not depend on the number of agents—persistent—the specification remains active until the user specifies a new command—and real time—the user can interact with and interrupt the swarm at any time. Our decentralized ergodic coverage algorithm enables the swarm to converge to user-specified targets, including in the setting of a large percentage of the agents unexpectedly failing. We demonstrated our system with simulated agents using command software developed by Raytheon BBN for the DARPA OFFSET program with up to 50 agents and also demonstrated our system working in the field, in real time, with a human operator in the loop for a couple of scenarios.

We note that using STOMP to interface with simulated agents requires substantial computation since the simulation is modeling every agent and their onboard computing. This practically makes simulation for hundreds of agents challenging; although this computational limitation is not present for a real physical swarm, where each agent is equipped with its own computational resource, it means that mission planning (which inevitably involves some simulation) is dependent on significant computational power. Future work includes distributing nodes across multiple computers and using more, and more powerful, CPUs to enable the simulation of hundreds or thousands of agents.

Field tests in the OFFSET FX-3 program revealed that the algorithms that we present here were capable of persistent coverage even in the presence of long-duration network loss. This practical result that was evident in the experiments themselves is not an obvious outcome for our algorithms; indeed, it is not clear why ergodic coverage may be insensitive to long-duration network dropouts. The authors' basic intuition is that coverage goals, specified by a particular distribution (either specified by a human operator or determined autonomously), have slower time constants than trajectory specifications. As a result, even though the trajectories may be executed at high speed, the rate at which the distribution representing coverage changes is at a much lower speed, leading to an overall low sensitivity to network failures. Future work includes analytically justifying this—including counterexample cases where this intuition fails.

In other future work, we are interested in creating a swarm that has closed-loop capability: automatically updating its target distribution based on the environmental stimuli that each agent receives (as opposed to simulating this capability by manually adding areas containing high and low expected information contents to the environment). Doing so would further reduce the burden on a human operator. We also want to investigate how to balance these automatic target distribution updates with updates made by the human operator. Furthermore, we would like to perform field tests that involve dynamic adversarial agents, such as persons wearing April Tags moving throughout the operating environment (such tests had originally been within the scope of the OFFSET field tests, but practical limitations and disruption due to the COVID-19 pandemic made them experimentally impractical).

REFERENCES

[1] G. Durantin, J.-F. Gagnon, S. Tremblay, and F. Dehais, "Using near infrared spectroscopy and heart rate variability to detect mental overload," *Behav. Brain Res.*, vol. 259, pp. 16–23, Feb. 2014.

[2] T. Gateau, C. P. C. Chanel, M.-H. Le, and F. Dehais, "Considering human's non-deterministic behavior and his availability state when designing a collaborative human-robots system," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 4391–4397.

[3] D. Lee, A. Franchi, P. R. Giordano, H. I. Son, and H. H. Bülthoff, "Haptic teleoperation of multiple unmanned aerial vehicles over the Internet," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 1341–1347.

[4] E. Tsykunov, R. Agishev, R. Ibrahimov, L. Labazanova, A. Tleugazy, and D. Tsetserukou, "SwarmTouch: Guiding a swarm of micro-quadrotors with impedance control using a wearable tactile interface," *IEEE Trans. Haptics*, vol. 12, no. 3, pp. 363–374, Jul. 2019.

[5] N. Ayanian, A. Spielberg, M. Arbesfeld, J. Strauss, and D. Rus, "Controlling a team of robots with a single input," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 1755–1762.

[6] J. Kato, D. Sakamoto, M. Inami, and T. Igarashi, "Multi-touch interface for controlling multiple mobile robots," in *Proc. Extended Abstr. Hum. Factors Comput. Syst.*, Apr. 2009, pp. 3443–3448.

[7] G. Podevijn, R. O'Grady, Y. S. G. Nashed, and M. Dorigo, "Gesturing at subswarms: Towards direct human control of robot swarms," in *Proc. Towards Auton. Robotic Syst. Conf. (TAROS)*, 2014, pp. 390–403.

[8] J. Nagi, A. Giusti, L. M. Gambardella, and G. A. D. Caro, "Human-swarm interaction using spatial gestures," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 3834–3841.

[9] L. H. Kim, D. S. Drew, V. Domova, and S. Follmer, "User-defined swarm robot control," in *Proc. Conf. Human Factors Comput. Syst.*, Apr. 2020, pp. 1–13.

[10] M. Micire, M. Desai, A. Courtemanche, K. M. Tsui, and H. A. Yanco, "Analysis of natural gestures for controlling robot teams on multi-touch tabletop surfaces," in *Proc. ACM Int. Conf. Interact. Tabletops Surf.*, Nov. 2009, pp. 41–48.

[11] C. Pinciroli and G. Beltrame, "Buzz: An extensible programming language for heterogeneous swarm robotics," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 3794–3800.

[12] M. A. Hsieh et al., "Adaptive teams of autonomous aerial and ground robots for situational awareness," *J. Field Robot.*, vol. 24, nos. 11–12, pp. 991–1014, Nov. 2007.

[13] S. McCammon et al., "Ocean front detection and tracking using a team of heterogeneous marine vehicles," *J. Field Robot.*, vol. 38, no. 6, pp. 854–881, Sep. 2021.

[14] N. E. Leonard, D. A. Paley, R. E. Davis, D. M. Fratantoni, F. Lekien, and F. Zhang, "Coordinated control of an underwater glider fleet in an adaptive ocean sampling field experiment in Monterey bay," *J. Field Robot.*, vol. 27, no. 6, pp. 718–740, Nov. 2010.

[15] A. Prabhakar et al., "Ergodic specifications for flexible swarm control: From user commands to persistent adaptation," in *Robotics: Science and Systems*. Corvallis, OR, USA, Jun. 2020.

[16] K. Giles and K. Giammarco, "Mission-based architecture for swarm composability (MASC)," *Proc. Comput. Sci.*, vol. 114, pp. 57–64, Jan. 2017.

[17] T. Balch and R. C. Arkin, "Behavior-based formation control for multirobot teams," *IEEE Trans. Robot. Autom.*, vol. 14, no. 6, pp. 926–939, Dec. 1998.

[18] T. Setter, H. Kawashima, and M. Egerstedt, "Team-level properties for haptic human-swarm interactions," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2015, pp. 453–458.

[19] A. Kolling, S. Nunnally, and M. Lewis, "Towards human control of robot swarms," in *Proc. 7th ACM/IEEE Int. Conf. Hum.-Robot Interact. (HRI)*, Mar. 2012, pp. 89–96.

[20] G. Bevacqua, J. Cacace, A. Finzi, and V. Lippiello, "Mixed-initiative planning and execution for multiple drones in search and rescue missions," in *Proc. Int. Conf. Automated Planning Scheduling*, Jun. 2015, pp. 315–323.

[21] P. Ogren, E. Fiorelli, and N. E. Leonard, "Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment," *IEEE Trans. Autom. Control*, vol. 49, no. 8, pp. 1292–1302, Aug. 2004.

[22] P. Song and V. Kumar, "A potential field based approach to multirobot manipulation," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2002, pp. 1217–1222.

[23] J. L. Baxter, E. K. Burke, J. M. Garibaldi, and M. Norman, "Multirobot search and rescue: A potential field based approach," in *Autonomous Robots and Agents*. Cham, Switzerland: Springer, 2007, pp. 9–16.

[24] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Trans. Robot. Autom.*, vol. 20, no. 2, pp. 243–255, Apr. 2004.

[25] M. Schwager, D. Rus, and J.-J. Slotine, "Decentralized, adaptive coverage control for networked robots," *Int. J. Robot. Res.*, vol. 28, no. 3, pp. 357–375, Mar. 2009.

[26] L. C. A. Pimenta, V. Kumar, R. C. Mesquita, and G. A. S. Pereira, "Sensing and coverage for a network of heterogeneous robots," in *Proc. 47th IEEE Conf. Decis. Control*, Dec. 2008, pp. 3947–3952.

[27] G. Lee, N. Y. Chong, and H. Christensen, "Tracking multiple moving targets with swarms of mobile robots," *Intell. Service Robot.*, vol. 3, no. 2, pp. 61–72, Apr. 2010.

[28] J. Cortes, S. Martinez, and F. Bullo, "Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions," *IEEE Trans. Autom. Control*, vol. 51, no. 8, pp. 1289–1298, Aug. 2006.

[29] C. Sampedro et al., "A flexible and dynamic mission planning architecture for UAV swarm coordination," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2016, pp. 355–363.

[30] D. Grecu and P. Gonsalves, "Agent-based simulation environment for UCAV mission planning and execution," in *Proc. AIAA Guid., Navigat., Control Conf. Exh.*, Aug. 2000, p. 4481.

[31] J. Alonso-Mora, E. Montijano, T. Nägeli, O. Hilliges, M. Schwager, and D. Rus, "Distributed multi-robot formation control in dynamic environments," *Auton. Robots*, vol. 43, no. 5, pp. 1079–1100, Jun. 2019.

[32] G. Swamy, S. Reddy, S. Levine, and A. D. Dragan, "Scaled autonomy: Enabling human operators to control robot fleets," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 5942–5948.

[33] P. Walker, S. A. Amraii, M. Lewis, N. Chakraborty, and K. Sycara, "Human control of leader-based swarms," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2013, pp. 2712–2717.

[34] M. A. Goodrich, S. Kerman, and S.-Y. Jung, "On leadership and influence in human-swarm interaction," in *Proc. AAAI Fall Symp., Hum. Control Bioinspired Swar*, 2012.

[35] S. L. Smith, M. Schwager, and D. Rus, "Persistent robotic tasks: Monitoring and sweeping in changing environments," *IEEE Trans. Robot.*, vol. 28, no. 2, pp. 410–426, Apr. 2012.

[36] R. N. Smith, M. Schwager, S. L. Smith, B. H. Jones, D. Rus, and G. S. Sukhatme, "Persistent ocean monitoring with underwater gliders: Adapting sampling resolution," *J. Field Robot.*, vol. 28, no. 5, pp. 714–741, Sep. 2011.

[37] A. Ghaffarkhah, Y. Yan, and Y. Mostofi, "Dynamic coverage of time-varying environments using a mobile robot—A communication-aware perspective," in *Proc. IEEE GLOBECOM Workshops (GC Wkshps)*, Dec. 2011, pp. 1297–1302.

[38] Y. Diaz-Mercado, S. G. Lee, and M. Egerstedt, "Distributed dynamic density coverage for human-swarm interactions," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2015, pp. 353–358.

[39] I. Abraham and T. D. Murphey, "Decentralized ergodic control: Distribution-driven sensing and exploration for multiagent systems," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 2987–2994, Oct. 2018.

[40] M. F. D. Olley, M. A. Peshkin, and J. E. Colgate, "Electronic controller haptic display with simultaneous sensing and actuation," U.S. Patent App. 15 117 631, Sep. 8, 2020.

[41] K. Naderi, J. Rajamäki, and P. Hämäläinen, "RT-RRT*: A real-time path planning algorithm based on RRT," in *Proc. 8th ACM SIGGRAPH Conf. Motion Games*, Nov. 2015, pp. 113–118.

[42] G. Mathew and I. Mezić, "Metrics for ergodicity and design of ergodic dynamics for multi-agent systems," *Physica D, Nonlinear Phenomena*, vol. 240, nos. 4–5, pp. 432–442, Feb. 2011.

[43] L. M. Miller, Y. Silverman, M. A. MacIver, and T. D. Murphey, "Ergodic exploration of distributed information," *IEEE Trans. Robot.*, vol. 32, no. 1, pp. 36–52, Feb. 2016.

[44] A. Mavrommati, E. Tzorakoleftherakis, I. Abraham, and T. D. Murphey, "Real-time area coverage and target localization using receding-horizon ergodic exploration," *IEEE Trans. Robot.*, vol. 34, no. 1, pp. 62–80, Feb. 2018.

[45] T.-R. Hsiang, E. M. Arkin, M. Bender, S. P. Fekete, and J. S. B. Mitchell, "Algorithms for rapidly dispersing robot swarms in unknown environments," in *Algorithmic Foundations of Robotics*. Cham, Switzerland: Springer, 2004.

[46] M. Day, L. Strickland, E. Squires, K. DeMarco, and C. Pippin, "Responding to unmanned aerial swarm saturation attacks with autonomous counter-swarms," in *Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR IX*. Orlando, FL, USA: SPIE, May 2018.

[47] W. Chen, J. Liu, H. Guo, and N. Kato, "Toward robust and intelligent drone swarm: Challenges and future directions," *IEEE Netw.*, vol. 34, no. 4, pp. 278–283, Aug. 2020.

● ● ●