A Game Benchmark for Real-Time Human-Swarm Control

Joel Meyer, Allison Pinosky, Thomas Trzpit, Ed Colgate, Todd D. Murphey

***Abstract*— We present a game benchmark for testing human-swarm control algorithms and interfaces in a real-time, high-cadence scenario. Our benchmark consists of a swarm vs. swarm game in a virtual ROS environment in which the goal of the game is to "capture" all agents from the opposing swarm; the game's high-cadence is a result of the capture rules, which cause agent team sizes to fluctuate rapidly. These rules require players to consider both the number of agents currently at their disposal and the behavior of their opponent's swarm when they plan actions. We demonstrate our game benchmark with a default human-swarm control system that enables a player to interact with their swarm through a high-level touchscreen interface. The touchscreen interface transforms player gestures into swarm control commands via a low-level decentralized ergodic control framework. We compare our default human-swarm control system to a flocking-based control system, and discuss traits that are crucial for swarm control algorithms and interfaces operating in real-time, high-cadence scenarios like our game benchmark. Our game benchmark code is available on Github; more information can be found at https://sites.google.com/view/swarm-game-benchmark.**

I. Introduction

An appealing aspect of robot swarms is their potential to be deployed into areas that are dangerous for humans to enter. Many dangers—like fire and unstable structures—cause physical harm and evolve rapidly. Operators controlling robot swarms deployed into environments with these dangers cannot assume the environment will remain constant; they need to be able to rapidly adapt to changing conditions, new information, and swarm agent dropout.

Many dangers are challenging to simulate because each environment is unique, however, simulations can still play an important role in testing human-swarm control algorithms and interfaces for these environments. Thus, the vision of sending human-swarm teams into dangerous, rapidly evolving scenarios faces two challenges: 1) many existing algorithms and interfaces are ill suited to the challenges of highly-dynamic environments and 2) there are no benchmarks (real-world or simulation-based) for testing such proposed algorithms and interfaces. To tackle these challenges, we present a game benchmark inspired by the challenges of real-time human-swarm control in high-cadence environments.

Even in benign scenarios, controlling swarms is challenging. As swarm size grows, the cognitive load on the operator increases, and it becomes difficult for an operator to task individual agents [1]. Prior work has sought to reduce the operator's cognitive load by autonomously planning agent trajectories via flocking algorithms [2], potential fields [3], fixed formations [4], linear temporal logic [5], collective motion [6] [7], and Voronoi partitions [8]. Unfortunately, many of these algorithms are too rigid for highly dynamic and dangerous scenarios and limit the variety of commands an operator can use to achieve a task.

Authors are with the Center for Robotics and Biosystems at Northwestern University. **Corresponding Author Email:** joelmeyer@u.northwestern.edu

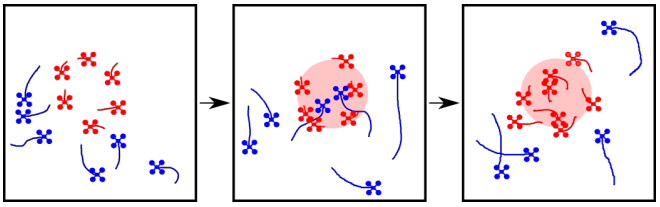

Fig. 1. **Swarm vs. Swarm Game:** Two players use interfaces to control opposing swarms in a shared, continuous space. The objective is to capture all agents on the opposing swarm by maneuvering one's own swarm to surround opposing agents. In this figure, as the blue swarm moves about, the red swarm forms a circular structure that surrounds the center of the game environment. Blue agents that pass through this surrounded area are captured by the red swarm according to the game rules. The game rules, which lend themselves to the emergence of vaguely Go-like structures during game play, are further explained in Section II-C). The blue agents that are captured become part of the red swarm, and can now be controlled by the red player.

Density-based swarm control algorithms, on the other hand, enable operators to specify flexible behavior to their swarm in dynamic and potentially dangerous scenarios. For example, authors in [9] developed a system that enabled human operators to control their swarm through density specifications via a touchscreen interface. Authors in [10] created an end-to-end swarm control system that leveraged their swarm's heterogeneous capability, used autonomously detected information to keep the swarm safe, and enabled operators to specify multimodal commands to their swarm via touchscreen that were adaptable in real-time. Furthermore, the touchscreen interfaces used in these works to send commands enabled both persistent swarm behavior and multiple variations of operator commands to achieve tasks through density specifications. Other work in swarm interfaces (e.g., touchscreen interfaces [11], brain-machine interfaces [12], swarm programming languages [13], and haptic control devices [14]), were more rigid in the types of behavior they could specify. Some of these interfaces were also not persistent and required the operator to constantly input commands to their system. In this work, we extend the system presented in [10] to demonstrate our game benchmark due to the system's promising real-time performance.

Prior work involving swarm testbeds and real-world swarm demonstrations has enabled researchers to test formation control, motion planning, and collision avoidance [15], [16], [17], [18], [19]. However, none of these testbeds or demonstrations were particularly dynamic nor did they assess high-cadence swarming. Work in game-theoretic scenarios involv-

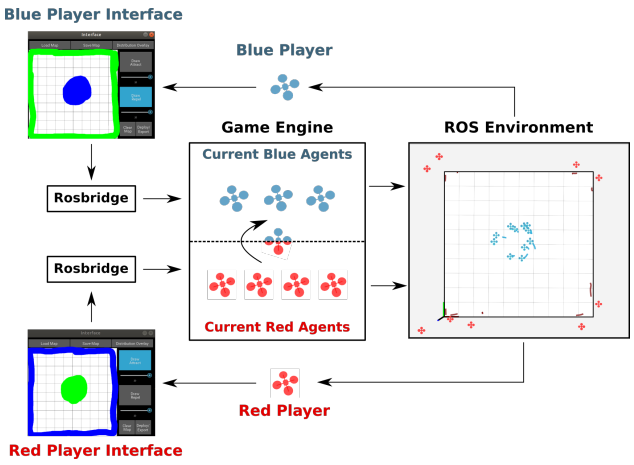

Fig. 2. **Overview of the Game Benchmark:** Two players use separate interfaces (pictured here are the touchscreen interfaces for our human-swarm control system described in Section III; researchers can choose to develop their own interfaces) to send commands via ROSbridge to their separate teams of agents. A ROS node tracks the state of the game (the positions of all agents) and determines whether an agent has been captured based on the rules of the game. Both players have their own displays (in RViz) showing the current state of the game. All code for the game benchmark will be made open source and can be executed on any laptop running Ubuntu 18.04 LTS with an Intel I5 or equivalent processor, and 4 GB of RAM. The code will contain a touchscreen interface for sending commands and an implementation of decentralized ergodic control for executing commands to enable researchers to demo the game benchmark "out of the box".

ing pursuit-evasion and racing with multi-agent systems [20], [21] [22], [23], [24] has created scenarios and algorithms that have approached what is needed to assess real-time swarm control in dynamic, high-cadence environments. The authors of [25] conducted a field test with unmanned aerial vehicles involving teams of 10 vs. 10 agents in which they examined swarm combat tactics in a mock aerial dog fight. However, all of these works were limited to team sizes under 10 agents, and some did not involve human operators.

Given the lack of systems and testing scenarios for real-time high-cadence human-swarm control, we have created a game benchmark (shown in Figure 1) that consists of a ROS-based virtual environment containing a swarm vs. swarm game. The swarm vs. swarm game targets scenarios that require evolving strategies—that is, rapid re-specification of strategies—in a rapidly changing environment (with respect to agent positions). We demonstrate our game benchmark with a human-swarm control system that uses a touchscreen interface to specify gesture-based objectives as distributions for robot swarms using real-time, decentralized ergodic control.

Section II describes our dynamic swarm vs. swarm game benchmark. Section III describes the default system for real-time human-swarm control we use to demonstrate our game. Section IV describes an example tactic that can be deployed in the game with our default system, highlights considerations operators may need to make when planning tactics in the game, compares our default system for demonstrating the game to a flocking-based method, and discusses the shortcomings of other swarm control systems for planning tactics in high-cadence scenarios. In Section V, we discuss traits of real-time human-swarm control algorithms and interfaces that may be desirable for high-cadence scenarios like our game benchmark and conclude with future work.

II. The Game Benchmark

In this section, we describe our benchmark, which is a dynamic game in which players control their swarms to capture *all* of the agents on the opposing team. Both teams start with the same number of agents, which is specified by the players when the game is initialized. The game is played until all the available agents are captured by one team (i.e., if the game starts with 20 agents total, 10 on each team, the game does not end until one team has all 20 agents and the other team has 0).

Players must strategically maneuver their swarm to surround agents on the opposing swarm while preventing their own agents from being captured—resulting in the emergence of vaguely Go-like structures (see Figure 1). We note, however, that our game is played in continuous time and space—there is no notion of teams "taking turns". Also, all of the agents in the game can be at any location in the environment at any time. Figure 2 shows the architecture for our game benchmark. The game benchmark architecture includes the agents, the control algorithm and interface the players choose to deploy, the virtual ROS environment, and the game engine dictating the capture rules.

A. The Agents

Our virtual agents are modeled as second order, 2-D point masses. Agents are spawned in one of four corners of the virtual ROS environment. Each agent is given a random initial altitude, distinct from all other agents in the environment, at which it remains for the entire game to avoid collisions. Agents subscribe to ROS topics that players publish commands to. In our demonstrated examples, each agent receives commands via touchscreen interface (discussed further in Section III-A), runs its own real-time, decentralized ergodic control algorithm (discussed further in Section III-B), and receives information from members of the team it is currently on. Each agent runs their controls at 10 Hz.

B. The Virtual ROS Environment

Our virtual ROS environment is rendered in a $[0,1]^2$ grid in RViz. The game visualization shows areas of the environment in which a player's swarm can capture agents on the opposing team. These areas are denoted by black shading on the background of the environment. A ROS game engine node enforcing the rules of the game (discussed in Section II-C) sends information to another ROS node that controls the game visualization. This information determines each agent's color (representing the team they are on) at each time step. The game visualization also renders a decaying trajectory history of each agent so players can see where members of the opposing swarm have recently been. The game visualization and game engine nodes both run at 30 Hz.

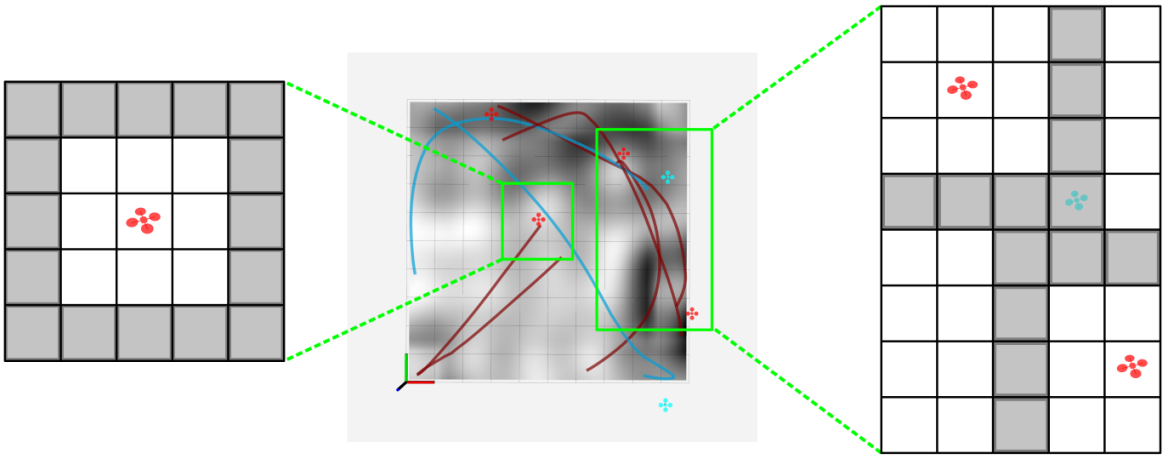

Fig. 3. **Swarm-based Game Geometry:** Each agent contributes to the time-averaged trajectories of the team it is on through its individual trajectory history. At each instance in time, the area surrounding an agent's current position is weighted as a capture area for its team (as seen in the pop-out on the left). Several agents with overlapping capture areas are more likely than an individual agent to capture an agent on the opposing team. The combined capture areas of the two red agents (visualized in the pop-out on the right) create a "net" and successfully capture the blue agent. Thus, commanding one's swarm to surround an opposing agent, or to trap an opposing agent along the sides of the environment, are more likely to successfully capture the opposing agent than driving one's swarm to be directly on top of the opposing agent. The dark shaded areas of the environment shown in the middle figure represent areas the red swarm can capture blue agents in. Darker shaded regions are more likely to lead to captures than lighter shaded regions.

C. Rules for Capturing Agents

The time-averaged trajectories of each swarm dictates areas of the environment in which opposing agents can be captured. The pop-out in Figure 3 shows how agents are captured. In order to create vaguely Go-like structures and motivate the concept of surrounding an opposing swarm, the time-averaged trajectories of the swarm over the whole environment (a `50x50` grid) are averaged over smaller neighborhoods represented by `5x5` sub-grids centered at each grid cell in the environment. Only the outer rows and columns of these sub-grids are used to calculate the "value" of each neighborhood.

Thus, regions of the environment that are completely surrounded by a swarm, or that are surrounded by a swarm and one or more of the four boundaries of the environment, have higher potential to capture opposing agents than areas of the environment a swarm is directly positioned over. At each time step in the game, if an opposing agent enters into a neighborhood containing a capture value greater than 75% of the maximum value over all neighborhoods, it will be captured. When an agent is captured, it immediately becomes controlled by the opposing player. The captured agent's command is updated to the last command published by the player controlling the new team it is now on. The captured agent also begins contributing to the areas its new team can capture opposing agents in through its trajectory history. If the new team the agent is on is running a decentralized control algorithm, the captured agent will also begin communicating with its teammates via ROS topics.

Players have a clear visual representation of where their swarm can capture opposing agents via the game visualization in RViz. Areas where their swarms are likely to capture opposing agents are shaded in black, while areas with a low or no chance of capture are shaded in white. The visual representation helps the players build intuition about the game rules. The density of one's own swarm has no bearing on its safety; a player must ensure the safety of their swarm through commands driving their swarm away from areas the opposing swarm is surrounding.

III. Our Default System for Real-Time Human-Swarm Control

This section describes our default human-swarm control system, which consists of a touchscreen interface and decentralized ergodic control framework we have developed for demonstrating our game benchmark. Our system is scale and permutation-invariant, which enables players to command their swarm in real-time and respond to changes in the environment by re-specifying behavior for their swarm.

A. Touchscreen Interface

Figure 4 shows our system's touchscreen interface for sending player commands to the swarm. We created our touchscreen interface using Kivy—an open source Python framework for developing graphical applications with touch capabilities. Our touchscreen interface works on any PC or tablet running Ubuntu 18.04 LTS or Windows 10 with Tkinter, Python3, and OpenCV. Players can control their swarm by specifying distributions with hand-gestures on a touchscreen tablet or with a mouse on PC; these distributions specify what their team of agents should do by interpreting the path of the gesture as a spatial distribution.

Players draw their distributions on a 2-D, top-down rendering of the virtual ROS environment. Players can select one of two input distribution types for areas of the environment: "Attract" (denoted in blue) which agents will converge to and

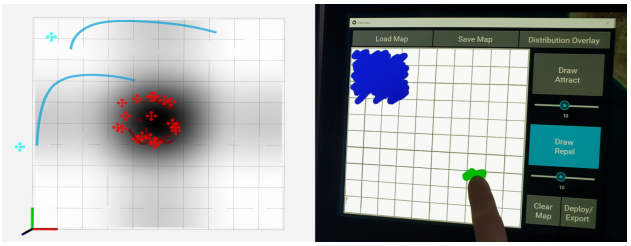

Fig. 4. **Touchscreen Interface:** The virtual ROS environment (left) next to our touchscreen interface with a target distribution drawn (right). Players can draw areas of high interest (which will attract the agents) in blue and areas of low interest (which will repel the agents) in green. When a player is satisfied with the target distribution they have drawn, they can send the target distribution to their team. Players can clear the map to draw a new distribution. Players can also modify an existing target distribution by drawing new targets on top of old targets. The touchscreen's drawing area directly maps to the virtual ROS environment's area. The darker shaded regions in the virtual ROS environment figure on the left correspond to areas in which the red swarm can capture members of the blue swarm (darker shaded regions are more likely to lead to captures than lighter shaded regions).

spend more time in, and "Repel" (denoted in green) which agents will avoid. New target distributions can be continually overlaid on top of previous ones, which enables quick updates to previous distributions. Players' target distributions are smoothed out with a Gaussian filter. They are then scaled in value according to the resolution and size of the virtual ROS environment before being sent via ROS websocket to the swarm agents.

B. Decentralized Ergodic Control

Player target distribution specifications provided via the touchscreen are transformed into low-level swarm commands through decentralized ergodic control. Decentralized ergodic control (described in detail in previous works [26], [27], [28]) provides a natural framework for specifying density-based swarm objectives, and enables a player to quickly and flexibly specify behavior for their swarm in response to the opposing swarm's behavior.

To define decentralized ergodic control, we start by introducing the dynamics of our system. Consider a set of N agents with state $x(t) = \left[x_1(t)^\top, x_2(t)^\top, \dots, x_N(t)^\top\right]^\top : \mathbb{R}^+ \to \mathbb{R}^{nN}$. From work in [26], we define the dynamics of the collective multi-agent system as

$$\begin{aligned}\dot{x} &= f(x,u) = g(x) + h(x)u \\ &= \begin{bmatrix} g_1(x_1) \\ g_2(x_2) \\ \vdots \\ g_N(x_N) \end{bmatrix} + \begin{bmatrix} h_1(x_1) & \dots & 0 \\ \vdots & \ddots & \\ 0 & & h_N(x_N) \end{bmatrix} u, \end{aligned} \quad (1)$$

where $g(x)$ is the free, unactuated dynamics of the multi-agent system and $h(x)$ is the system's dynamic control response. We seek to find a set of controls u for the multi-agent system that minimizes the system's ergodic metric $\mathcal{E}$ with respect to some player target specification $\phi(s)$ where $s \in \mathbb{R}^2$ is a point in the game environment.

The ergodic metric $\mathcal{E}$ provides a way to calculate the "difference" between player target specifications $\phi(s)$ and past agent trajectories [28] (which are represented as c_k values using Fourier decompositions)—similar to the Kullback-Leibler divergence for comparing two distributions.

We can use both Fourier decompositions to calculate the ergodic metric $\mathcal{E}$ (introduced in [28]):

$$\mathcal{E}(x(t)) = q \sum_{k \in \mathbb{N}^\nu} \Lambda_k (c_k - \phi_k)^2 \quad (2)$$

where $q \in \mathbb{R}^+$ is a scalar weight, $\Lambda_k = (1 + ||k||^2)^{\frac{\nu+1}{2}}$ is a weight on the frequency coefficients, c_k is the Fourier decomposition of the agents' trajectories in a player's swarm over a fixed time horizon, and ϕ_k is the Fourier decomposition of the player's target specification $\phi(s)$ [28].

We can determine the ergodic metric's sensitivity to different control inputs u by differentiating the ergodic metric with respect to a control application duration λ at some optimal application time τ. This results in the costate equation

$$\dot{\rho} = -\mathcal{E}(x(t)) \frac{\partial F_k}{\partial x} - \frac{\partial \Phi}{\partial x} - \frac{\partial f}{\partial x} \rho(t) \quad (3)$$

where Φ is the exponential control barrier function that keeps the agents in the operating environment, F_k is the cosine basis function, and $f(x,u)$ is the system dynamics.

We can then write an unconstrained optimization problem

$$J = \int_{t_i}^{t_i+T} \left.\frac{\partial \mathcal{E}}{\partial \lambda}\right|_\tau + \frac{1}{2} ||u_\star(t) - u_{\text{def}}(t)||_R^2 \, dt \quad (4)$$

where R is a positive definite matrix that weights the control, $u_\star$ is the optimal control and u_{def} is some default control. The default control u_{def} could be that the agent moves forward at its current velocity. In this paper, u_{def} is zero. The control that minimizes this objective J is:

$$u_\star(t) = -R^{-1} \frac{\partial f}{\partial u}^T \rho(t) + u_{\text{def}}(t)$$

which is calculated and applied at every time step to the player's team of agents for the player's current target specification input.

C. Scale and Permutation-Invariance

Our touchscreen interface and decentralized ergodic control algorithm enable our system to be scale and permutation-invariant with respect to player target distributions. Figure 5 shows swarms containing 6, 12, and 24 agents responding to three different player target distributions specified through the touchscreen interface. The combination of our touchscreen interface and decentralized ergodic control algorithm produces scale-invariant swarm behavior, since all three swarm sizes converge to the three different target distributions. Our system also produces permutation-invariant swarm behavior, since the swarm will converge to the player's target distribution from different permutations of agent positions (i.e., the player's swarm contains 10 agents that are in different positions when the player specifies the target. If Agent 1, Agent 2, etc. swapped positions, the swarm would still converge to the player's target).

Our system's scale and permutation-invariance enables the player to plan swarm-level behavior instead of individual

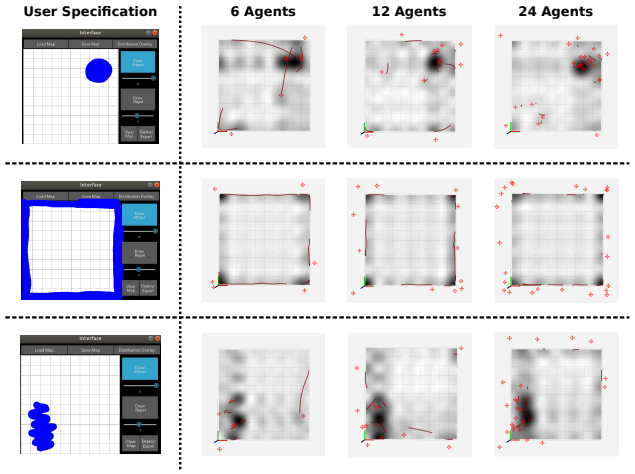

Fig. 5. **Specification is Independent of Swarm Size with our Default Human-Swarm Control System:** In this example, the black and white shaded areas of the environment represent the time-averaged trajectories of the swarm (not capture areas based on the game rules), with black areas corresponding to areas the player's swarm's trajectories have spent more time in relative to white areas. The player's specifications are scale-invariant, since all three swarm sizes converge to the player target specifications. The player target specifications are also permutation-invariant, since the swarm will converge to the player's target specification from different permutations of the same set of agent positions.

agent trajectories. For high-cadence scenarios like our game benchmark, as the number of agents the player controls increases and the time horizon the player has to plan decreases, the player cannot think strategically in terms of individual agents and their trajectories. Our system enables players to make strategic decisions based on the general positions and densities of their own swarm and their opponent's swarm.

Furthermore, the decentralized nature of our ergodic control algorithm enables players to maintain their strategy (the most recent target distribution they sent to their swarm) regardless of how many agents are currently under their control. Even if the number of agents under their control is fluctuating, their swarm will still converge to their last target. These traits are advantageous for players planning tactics in high-cadence scenarios like our game benchmark. We elaborate on these traits in Section IV.

IV. Swarm Tactics

In this section, we describe an example ensemble tactic a player can deploy with our default human-swarm control system to beat an opponent in the game benchmark. We then discuss tactical considerations players may need to make during the game. We also discuss the shortcomings of other methods for human-swarm control and the challenges players may face when using these other methods to plan tactics in high-cadence scenarios like our game benchmark.

A. Example Game Tactics

Figure 6 shows an example sequence of target specifications a player can use with our default human-swarm control system to capture agents on the opposing team. The nature of our game enables a player to quickly capture all

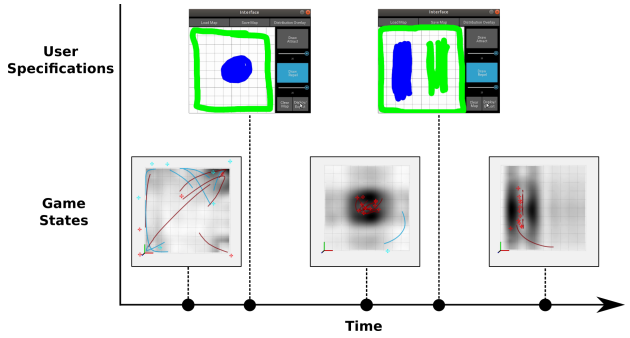

Fig. 6. **Ensemble Game Tactics:** In the game state on the far left, it is difficult to discern any particular structure. Once the red player specifies a unimodal target distribution, the red agents converge to the center of the environment and capture opposing blue agents in the process. The red player then specifies their red agents to track the remaining blue agent along the left side of the environment to capture it and win the game. The dark shaded areas in the game state figures correspond to areas in which the red team can capture blue agents. The darker shaded regions are more likely to lead to captures than the lighter shaded regions.

the opposing agents (and win the game) with a well-timed sequence of commands. In the initial game state shown in Figure 6 on the far left, it is difficult to discern any particular structure in the agent positions. Once the red player specifies a unimodal distribution in the center of the environment with our interface, the red agents converge to a visible structure via decentralized ergodic control and capture opposing blue agents in the process. The red player then specifies their swarm to track the remaining blue agent as it circles along the left side of the environment. The red swarm then captures the blue agent and wins the game.

This is one of many possible winning specification sequences players can make with our human-swarm control system (see multimedia attachment). Players can reason about how they can use the distribution of their swarm in the environment to create the most opportunities to capture opposing agents. Players can also reason about the current position of the opposing swarm and attempt to predict where the opposing swarm will be in subsequent time steps.

B. Player Tactic Considerations

Since the rules of the game allow any agent to be captured regardless of how many of its team members are surrounding it, players might be wary of sending commands to their swarm that cause all of their agents to be in positions close to one another. While agents on the same team that are close together create areas of the environment that are more likely to capture agents on the opposing team (due to overlapping capture areas from the game rules), the opponent may maneuver their swarm in a way that enables them to simultaneously capture all the agents on the player's team and win the game. Thus, there are consequences to every maneuver a player makes; a player may maneuver to capture a small number of agents on the opposing team, only to find that their agents have now been steered into an area where they can be captured.

The dynamic, high-cadence nature of the game also re-

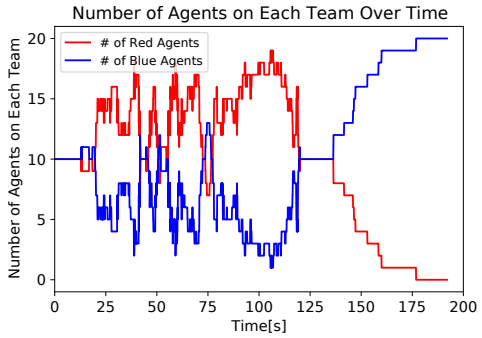

Fig. 7. **Swarm Size Changes Rapidly During Game Play:** This figure shows the number of agents on each team over the course of an example game between two players. The game starts at approximately the 15s mark and ends right before the 200s mark. The number of agents on each team changes rapidly over the course of the game. Near the end of the game, the frequency of agent captures stabilizes as the blue player accumulates more total agents than the red player. The blue player eventually captures all the red agents and wins the game.

quires the player to consider the amount of time they take to send commands to their swarm. Intricate commands (such as drawing detailed target specifications with our default human-swarm control system) may require the player to solely focus on creating their command (i.e., focusing on their touchscreen interface while they draw) instead of looking at the rapidly changing swarm positions in the environment. Figure 7 shows how the number of agents on each team fluctuates over the course of an example game. As seen in the figure, team sizes change rapidly over short periods of time. A brief advantage in number of agents under a player's control can quickly vanish in a matter of seconds. This rapid cadence may also affect player strategy in that players with larger swarms may be able to study the environment for emerging patterns in an opponent's play and take longer to specify new commands for their swarm since they can afford to lose agents without losing the game. Players down to the last member of their swarm may instead have to repeatedly specify commands that cause their agent to quickly move around the environment to create opportunities to capture opposing agents and rebuild the size of their swarm.

We note here that it *is possible* to recover from a dramatic ratio in numbers of agents between teams. One agent, controlled strategically, can be sufficient to build up team size again. Players may find that they need to play many games before they become adept at reasoning about their swarm's behavior as a whole. A naive player may find that even an opponent swarm specified by a uniform coverage command across the domain—with no changes over time—is a challenging adversary.

Various strategies for trapping and capturing agents on the opposing team may emerge for players from repeated game play. Player tactics may mirror some of those presented in other work in pursuit-evasion, such as [23], or in swarm tactics work such as [25]. In some instances, players may find it useful to make the environment more chaotic by having their swarm spread out in all directions, which could disorient the opponent. For instance, players using our default human-swarm control system could crash their swarm into one side of the environment through "wavefront" attacks via specifically-drawn attraction and repulsion regions, which may quickly capture many opposing agents at high risk to the player losing their own agents.

C. Challenges of Applying Traditional Swarm Control Methods to Game Tactics

Leader-follower methods such as those presented in [29], [30] may not be able to perform the tactics described above because pre-designated leaders (or leaders elected at each time step) could be captured. Each team would have to re-appoint a leader from their available agents, which may not be possible with certain leader-election algorithms or methods that involve multiple leaders [31] at high-cadence. Likewise, influencing the swarm by having the player take control of an individual agent or sub-team (such as in [32]) may also be infeasible since the agent the player is controlling could be captured. It may also be difficult for the player to select an individual agent or sub-team in a high-cadence environment due to cognitive load.

Formation-based control methods, such as those presented in [4], [33], [34], may limit how a player can specify different locations for sub-groups of their swarm to converge to for strategic purposes. It is not clear how the same multimodal maneuvers performed by drawing density specifications with our default system can be achieved with swarm formation control methods that may require fixed formations to be determined beforehand. Some of these formation-based methods that are non-decentralized may also be affected by communications failure or agents changing teams.

Both leader-follower and formation-based control methods, however, do have advantages over our default system in the different types of automatic swarm-response behavior they can enable. For instance, leader-follower methods could enable a player to create automatic multi-modal behaviors for elected leaders in their swarm to perform in response to different opposing swarm configurations during game play. Formation-based methods could enable a player to create a library of swarm formations they found useful for strategic purposes that they could then deploy at any instant during the game. Such automatic behaviors would not be possible with our default system (in its current form).

Figure 8 compares our default method for human-swarm control to a flocking-based control algorithm (adapted from work in [34]). In the figure, the player specifies a bimodal target for their swarm to converge to. Our system enables players to specify both targets at once, while flocking requires both targets to be specified in sequence, one at a time (denoted by the circles in the figure). The desired "compactness" of the player's swarm at each target would also have to be specified through an attractor weight parameter (green color saturation). On the other hand, our system enables the player to draw a larger region for more diffuse coverage and a smaller region for more compact coverage

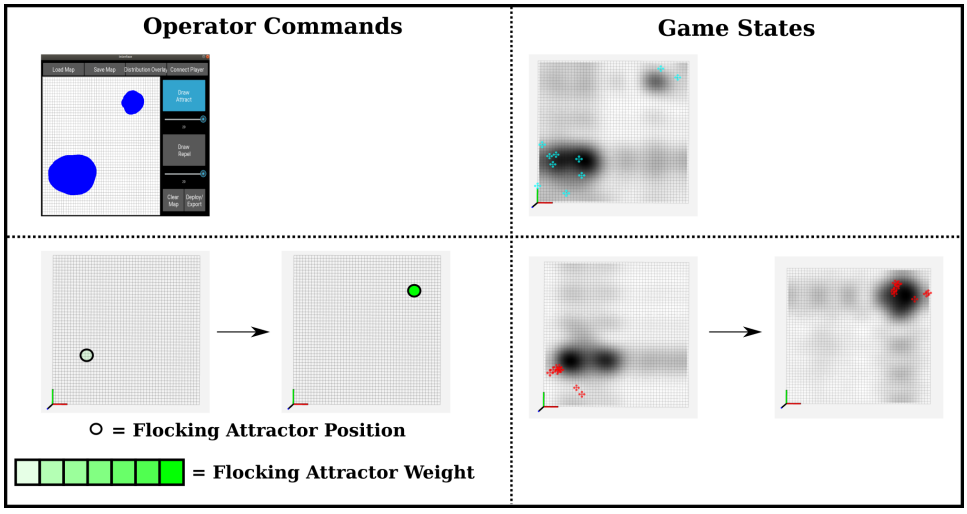

Fig. 8. **Our Default Method for Human-Swarm Control vs. Flocking:** This figure compares how our system for human-swarm control and a flocking-based method converge to a bimodal target (with a more diffuse target in the bottom left corner and a more compact target in the upper right). The top row shows the results for our default system for human-swarm control. The player specifies the bimodal target all at once, drawing a larger circle on their interface for the more diffuse target and a smaller circle for the more compact target. The bottom row shows the results for the flocking-based method. The player has to specify each of the targets in a sequence instead of all at once. The player also has to specify both the flocking attractor positions (denoted by circles) and the attraction weights (denoted by green color saturation) for each of these targets to achieve the desired level of compactness.

in a single specification. Although our default system can perform some tactical maneuvers (such as the one above) in fewer specifications than the flocking-based method, the flocking-based method is still scale and permutation-invariant (like our default system) and may prove superior in other scenarios. For instance, specifying a single attractor position and weight to capture opposing agents in a particular area of the game environment may take a player less time than drawing an attractor region with our touchscreen interface.

Thus, while the system we have used to demonstrate the game benchmark contains many advantages over more traditional methods for swarm control, we only make those arguments in specific cases of specific algorithms; other researchers implementing their own systems for real-time human-swarm control in this game benchmark will lead to better comparisons and faster development of capable algorithms. The game benchmark is a useful opportunity for the swarm and human-robot interaction communities to determine what traits are necessary for these systems to have to succeed in high-cadence scenarios. We conclude with what some of these traits are in the next section.

V. Discussion

We have designed a game benchmark for assessing human-swarm control algorithms and interfaces in a real-time, high-cadence scenario. We demonstrated our game benchmark scenario using a default human-swarm control system that was scale and permutation-invariant. We provided discussion on example tactics players can employ, tactical considerations players may need to make while playing our game benchmark, and compared our default system to other methods for human-swarm control.

Based on our demonstrations, scale and permutation-invariance are important characteristics for algorithms and interfaces operating in high-cadence scenarios like our game benchmark. It is difficult to deploy tactics at high-cadence without algorithms and interfaces that scale to teams of different sizes and enable players to specify locations for their swarm to converge to without having to keep track of unique agent locations or which agents have been captured or newly acquired. Instead of reasoning about individual agents, players must be able to reason about their and their opponent's swarms as a whole.

A specific advantage of our default system for human-swarm control is that the touchscreen interface does not require player swarm specifications to be pre-determined. Different players can draw different distributions on the touchscreen that represent the same "tactic" at a higher level (i.e., trapping opposing agents by drawing some type of shape). Players who are interested in using techniques from machine learning to learn possible tactics (that humans could deploy) for swarm systems operating in high-cadence scenarios may want to develop control algorithms and interfaces (with or without touch-based modalities) that afford machine learning agents the same specification flexibility as our system during the learning process. Scale and permutation-invariance may also be necessary traits for these algorithms and interfaces to make tactic learning feasible (especially if techniques from reinforcement learning are used).

Future work could develop a virtual adversary to compete against human players in this benchmark scenario. We envision this adversary as an opponent benchmark that other researchers could use to test their real-time human-swarm control systems against. Then, the performance of real-time human-swarm control systems could be compared via

number of wins and duration of game-play. Finally, the game benchmark could be extended to conduct human subject testing to empirically determine what types of strategies human players employ. Biometric data could be collected from players as they play the game to determine how much cognitive load they are experiencing and if certain changes to the human-swarm control algorithm or interface produce more or less cognitive load.

ACKNOWLEDGMENTS

This material is based on work supported by the United States National Science Foundation grant CNS 1837515 and by the Defense Advanced Research Projects Agency (DARPA) OFFSET SPRINT grant HR00112020035. The views, opinions, and/or findings expressed are those of the authors, and should not be interpreted as representing the views or policies of either agency. Author Joel Meyer was supported by a National Defense Science and Engineering Graduate Fellowship.

REFERENCES

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

[2] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: algorithms and theory," *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, Mar. 2006.

[3] A. Howard, M. J. Matarić, and G. S. Sukhatme, "Mobile Sensor Network Deployment using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem," in *Distributed Autonomous Robotic Systems 5*, 2002, pp. 299–308.

[4] N. Michael and V. Kumar, "Planning and Control of Ensembles of Robots with Non-holonomic Constraints," *The International Journal of Robotics Research*, vol. 28, no. 8, pp. 962–975, Aug. 2009.

[5] J. Chen, R. Sun, and H. Kress-Gazit, "Distributed Control of Robotic Swarms from Reactive High-Level Specifications," in *IEEE 17th International Conference on Automation Science and Engineering (CASE)*, Aug. 2021.

[6] K. Szwaykowska, L. M.-y.-T. Romero, and I. B. Schwartz, "Collective Motions of Heterogeneous Swarms," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 810–818, July 2015.

[7] H. Zhao, H. Liu, Y.-W. Leung, and X. Chu, "Self-Adaptive Collective Motion of Swarm Robots," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 4, pp. 1533–1545, Oct. 2018.

[8] C. Sampedro, H. Bavle, J. L. Sanchez-Lopez, R. A. S. Fernández, A. Rodríguez-Ramos, M. Molina, and P. Campoy, "A flexible and dynamic mission planning architecture for UAV swarm coordination," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2016.

[9] Y. Diaz-Mercado, S. G. Lee, and M. Egerstedt, "Distributed dynamic density coverage for human-swarm interactions," in *2015 American Control Conference (ACC)*, July 2015.

[10] A. Prabhakar, I. Abraham, A. Taylor, M. Schlafly, K. Popovic, G. Diniz, B. Teich, B. Simidchieva, S. Clark, and T. Murphey, "Ergodic Specifications for Flexible Swarm Control: From User Commands to Persistent Adaptation," *Robotics: Science and Systems*, June 2020.

[11] 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 Transactions on Haptics*, vol. 12, no. 3, pp. 363–374, July 2019.

[12] G. K. Karavas, D. T. Larsson, and P. Artemiadis, "A hybrid BMI for control of robotic swarms: Preliminary results," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept. 2017.

[13] C. Pinciroli and G. Beltrame, "Buzz: An extensible programming language for heterogeneous swarm robotics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2016.

[14] 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 *IEEE International Conference on Robotics and Automation*, May 2011.

[15] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2017.

[16] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The GRASP Multiple Micro-UAV Testbed," *IEEE Robotics Automation Magazine*, vol. 17, no. 3, pp. 56–65, Sept. 2010.

[17] D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. Ames, E. Feron, and M. Egerstedt, "The Robotarium: A remotely accessible swarm robotics research testbed," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2017.

[18] B. Kate, J. Waterman, K. Dantu, and M. Welsh, "Simbeeotic: A simulator and testbed for micro-aerial vehicle swarm experiments," in *ACM/IEEE 11th International Conference on Information Processing in Sensor Networks (IPSN)*, Apr. 2012.

[19] T. H. Chung, M. R. Clement, M. A. Day, K. D. Jones, D. Davis, and M. Jones, "Live-fly, large-scale field experimentation for large numbers of fixed-wing UAVs," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2016.

[20] M. Wang, Z. Wang, J. Talbot, J. C. Gerdes, and M. Schwager, "Game-Theoretic Planning for Self-Driving Cars in Multivehicle Competitive Scenarios," *IEEE Transactions on Robotics*, vol. 37, no. 4, Aug. 2021.

[21] K. Shah and M. Schwager, "Multi-agent Cooperative Pursuit-Evasion Strategies Under Uncertainty," in *Distributed Autonomous Robotic Systems*, 2019.

[22] ——, "GRAPE: Geometric Risk-Aware Pursuit-Evasion," *Robotics and Autonomous Systems*, vol. 121, Nov. 2019.

[23] A. Pierson, Z. Wang, and M. Schwager, "Intercepting Rogue Robots: An Algorithm for Capturing Multiple Evaders With Multiple Pursuers," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 530–537, Apr. 2017.

[24] A. Pierson and M. Schwager, "Controlling Noncooperative Herds with Robotic Herders," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 517–525, Apr. 2018.

[25] 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*, May 2018.

[26] I. Abraham and T. D. Murphey, "Decentralized Ergodic Control: Distribution-Driven Sensing and Exploration for Multiagent Systems," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, Oct. 2018.

[27] A. Mavrommati, E. Tzorakoleftherakis, I. Abraham, and T. D. Murphey, "Real-Time Area Coverage and Target Localization using Receding-Horizon Ergodic Exploration," *IEEE Transactions on Robotics*, vol. 34, no. 1, pp. 62–80, Aug. 2017.

[28] G. Mathew and I. Mezić, "Metrics for ergodicity and design of ergodic dynamics for multi-agent systems," *Physica D: Nonlinear Phenomena*, vol. 240, no. 4-5, pp. 432–442, Feb. 2011.

[29] M. A. Goodrich, S. Kerman, and S.-Y. Jung, "On Leadership and Influence in Human-Swarm Interaction," *Association for the Advancement of Artificial Intelligence (AAAI)*, p. 6, 2012.

[30] P. Walker, S. Amirpour Amraii, N. Chakraborty, M. Lewis, and K. Sycara, "Human control of robot swarms with dynamic leaders," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept. 2014.

[31] P. Walker, S. Amirpour Amraii, M. Lewis, N. Chakraborty, and K. Sycara, "Control of swarms with multiple leader agents," in *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct. 2014.

[32] G. Swamy, S. Reddy, S. Levine, and A. D. Dragan, "Scaled Autonomy: Enabling Human Operators to Control Robot Fleets," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2020.

[33] S. Hauri, J. Alonso-Mora, A. Breitenmoser, R. Siegwart, and P. Beardsley, "Multi-Robot Formation Control via a Real-Time Drawing Interface," in *Field and Service Robotics*, 2014, vol. 92, pp. 175–189.

[34] S. Haubert, S. Leven, M. Varga, F. Ruini, A. Cangelosi, J.-C. Zufferey, and D. Floreano, "Reynolds Flocking in Reality with Fixed-Wing Robots: Communication Range vs. Maximum Turning Rate," in *IEEE/RSJ International Conference on Robots and Systems*, Sept. 2011, pp. 5015–5020.