Special Issue: Dynamic Large-Scale Swarm Systems in Urban Environments: Results from the DARPA OFFSET Program

Field Report

Immersive Interaction Interface (I3): A Virtual Reality Swarm Control Interface

Phillip Walker, Joshua Hamell, Christopher Miller, Jack Ladwig, Helen Wauck and Peter Keller
SIFT, LLC, Minneapolis, MN 55401

Abstract: The Defense Advanced Research Projects Agency (DARPA) OFFensive Swarm-Enabled Tactics (OFFSET) program seeks to develop swarms of up to 250 aerial- and ground-based platforms to aid small-unit infantry forces to accomplish missions in complex urban environments. Over the course of four years, the OFFSET performers regularly tested and improved the autonomy, hardware, communications infrastructure, and logistics necessary to enable such a vision. In this paper, we present the Immersive Interaction Interface (I3), a virtual reality interface designed by Smart Information Flow Technologies (SIFT), as part of the BBN/Raytheon-led Command and Control of Aggregate Swarm Tactics (CCAST) team. I3 served as the main control interface for the team throughout the program, and this paper discusses the primary innovative features of I3 as compared to prior swarm control interfaces, how I3 was used in the field during exercises, and lessons learned over the course of the program. The paper also presents numerical results from the final three field exercises of the OFFSET program, demonstrating how a single operator used I3 effectively to control large numbers of unmanned vehicles.

Keywords: aerial robotics, cooperative robots, human robot interaction

1. Introduction

For years, researchers and engineers have sought to bring swarms of robots into real world environments. Swarms are large teams of coordinating autonomous or semiautonomous vehicles that seek to solve a problem difficult for humans or small robot teams to handle: accomplishing goals requiring coordinated sensing and actuation distributed out over a large geographic area. Developers of swarms also tout other benefits—that swarms are robust to the failure of individual members (Winfield and Nembrini, 2006; Mulgaonkar et al., 2017), that new members can be added easily to scale the swarm up to larger sizes (Şahin, 2004), and that the coordinated interaction amongst swarm members can give rise to emergent, often beneficial, aggregate behaviors (Sharkey, 2006; Winfield et al., 2005).

Those studying swarm algorithms and hardware in simulation and laboratory settings oftentimes develop solutions that provide theoretical guarantees of certain behaviors or outcomes provided

Received: 17 March 2022; revised: 14 October 2022; accepted: 6 February 2023; published: 13 April 2023.

Correspondence: Phillip Walker, SIFT, LLC, Minneapolis, MN 55401, Email: pwalker@sift.net

DOI: https://doi.org/10.55417/fr.2023019

certain assumptions about sensing and other hardware capabilities are met. Additionally, new human-swarm interfaces demonstrate how a single operator, or small number of operators, can meaningfully control a swarm performing a myriad of tasks in simulated environments. However, as large swarms of robots begin to move from controlled laboratory settings and simulations to real world environments, much of what we have learned and assumed about swarms comes into question. Take for instance the robustness benefit described above—that swarms can continue to operate successfully even as individual members fail. Throughout the OFFSET program, we observed on multiple occasions how individual failures can significantly impact the success of the mission, most notably by causing significant added stress and required attention on the part of the swarm operator. The interfaces we design to control swarms necessarily must evolve from the current state-of-the-art to account for the uncertain and unreliable nature the real world presents.

The Defense Advanced Research Projects Agency (DARPA) OFFensive Swarm-Enabled Tactics program, hereafter referred to as OFFSET, seeks to address the problems inherent in bringing swarms into real world environments[1]. Over the course of four years, the two OFFSET teams fielded up to 250 platforms in urban environments with the goal of executing increasingly complex intelligence, surveillance, and reconnaissance (ISR) missions. Along with significant challenges in hardware, autonomy, and logistics, this goal presented a novel problem in the human factors and interface design space as well: how to design an interface that could provide meaningful control of these swarms to a single operator and allow them to handle and overcome hardware failure, noisy sensor data, and evolving missions over the course of multiple hours.

This paper presents the development and findings of the CCAST team's Immersive Interaction Interface (I3), a virtual reality game-like interface for controlling swarms in urban environments using the broader CCAST system (Clark et al., 2021). I3 provides a single operator multiple levels of control—from individual robot commands to high-level swarm tactics—while maintaining a high level of situational awareness that traditional 2D screen-based interfaces often struggle to provide.

In Section 2, we provide an overview of existing research in swarms and human-swarm interfaces. Section 3 walks through the primary features of I3 required for the OFFSET exercises. Section 4 describes how an operator used I3 during a typical OFFSET exercise, and reports on quantitative metrics from the final three field exercises (FXs). Section 5 discusses the primary points of difficulty when using I3 over the course of the program, how we attempted to address those difficulties through further development, and what lessons were learned over the course of development. Finally, Section 6 concludes the paper and describes what problems remain for future work to address.

2. Related Work

2.1. Attention and Control Complexity

In the field of computer science, researchers use the notion of computational complexity to characterize the time it takes for an algorithm to solve a problem as a function of the size of its input. For example, algorithms that scale linearly with input size are described as having $O(n)$ complexity, while algorithms that remain constant as input changes, or scale with the square of the input size, are $O(1)$ or $O(n^2)$, respectively. In (Lewis et al., 2006), the authors apply this idea to the field of human robot interaction to determine the *cognitive complexity* of the operator's required effort to control a human-swarm system. If an operator is controlling n independent robots by teleoperation, then the cognitive complexity of the system is $O(n)$, because each new robot requires a linear increase in the operator's attention or interaction time. If instead, the same group of robots are coordinating, and the operator must control those interactions as well as individual teleoperation, the complexity would be higher—likely $O(n^2)$. The idea behind swarms is that the cognitive complexity should be sublinear, perhaps even $O(1)$. Here, autonomy onboard the robots

[1] To see video from the OFFSET field exercises, which includes some footage of I3 in use, see: https://www.youtube.com/watch?v=km0LWvnMrtEandhttps://www.youtube.com/watch?v=W34NPbGkLGI.

handle the interactions as well as the low-level decisions and actions required to complete the mission. The operator attends only to the overall goal of the swarm; in effect, the operator treats the swarm as a single entity.

The fan-out model, proposed in (Olsen Jr and Wood, 2004), introduces the concept of *neglect tolerance* to multi-robot systems. Neglect tolerance is the assumption that every robot can operate fully autonomously, with no human input, for some length of time (the neglect time) before they require the operator's attention again. For example, the operator might provide a destination or series of waypoints for the robot to follow and, depending on the underlying autonomy of the robot, can be assured that the robot will be occupied for some length of time until it reaches the goal and again requires new input. Newer models seek to better formalize operator attention in multi-robot and swarm control (Morris et al., 2013), develop and study scheduling schemes for operators (Chien et al., 2011; Chien et al., 2012), and model operator trust in multi-robot teams (Mahani et al., 2020; Nam et al., 2019).

Although human-swarm interaction entails some of the same problems, the different nature of swarms when compared to single robots or multi-robot systems presents different problems that require new solutions to address. For instance, in situations where the operator is treating the entire swarm as a single entity, neglect tolerance and the fan-out model no longer apply. In fact, researchers have observed that in some cases the opposite is true. In (Walker et al., 2012), the authors present the concept of *neglect benevolence*—the idea that due to the distributed nature of many swarm algorithms, operators might **benefit** from ignoring the swarm for some length of time before issuing new commands. To do the opposite would require the swarm to restabilize, for instance to converge on a new goal, and degrade performance overall. In (Nagavalli et al., 2014; Nagavalli et al., 2015), the authors prove and more formally model this concept. In many ways, OFFSET makes use of both swarms and multi-agent systems depending on the stage of scenario execution and how many working vehicles the I3 operator has under their control. Therefore an operator must take both neglect tolerance and neglect benevolence into account—neglect tolerance when dealing with individual robots, often in the later stages of mission, and neglect benevolence when dealing with larger surveillance tasks with numerous aerial vehicles, often early in the mission. See Section 4 for more details.

Another issue presented by remote control of multi robot systems, and especially swarms, is in handling suboptimal communication environments and infrastructure. When a task requires single operator control of a swarm, the communication between the swarm and the operator becomes of particular concern. Designers of swarm algorithms borrow much from past research on wireless sensor networks (WSNs), see (Rashid and Rehmani, 2016) for a survey of WSNs in environments and applications that often match those of swarms. For example, some interfaces for swarm control attempt to mitigate some of the communication restrictions by creating summary displays when bandwidth is low and data is limited (Nunnally et al., 2012) or predictive displays when latency in the network is high (Walker et al., 2012).

2.2. Control Algorithms for Swarm Robotics

The majority of early work on swarm control focuses on biologically inspired algorithms with human input to bias the natural convergence of these algorithms. Flocking, based on (Reynolds, 1987) and later extended by (Couzin et al., 2002), is a common approach to moving swarms between two locations. Swarm operators achieve goal-directed flocking either by biasing the *alignment* vector of each robot performing a traditional flocking algorithm (Walker et al., 2012), or by controlling leaders within the swarm to implicitly bias neighbors (Goodrich et al., 2012; Liu and Gao, 2020). Furthermore, modifying the weights and boundaries of the three zones used by flocking algorithms (repulsion, alignment, and cohesion) allows operators to force the swarm to rendezvous or disperse, and generally achieve a wider range of swarm capabilities (Couzin et al., 2002; Walker, 2017). Other approaches more explicitly trigger changes in swarm behavior by sending a signal requesting the swarm switch between different predefined algorithms (Kira and Potter, 2009; Kolling et al., 2012).

The bio-inspired paradigm is far from the only one used as a method for designing algorithms for swarm control. Other approaches include game theory (Jang et al., 2018a), which considers the problem of decision-making and task allocation in swarms as a hedonic game; control theory (Gazi and Fidan, 2006; Bullo et al., 2009; Jang et al., 2018b); markov chains (Bandyopadhyay et al., 2017); and even more esoteric control schemes like species-based task allocation (Prorok et al., 2017) or pheromone-based policies (Sauter et al., 2008; Sauter et al., 2009). While the focus of this paper is on the virtual reality interface for viewing a swarm and providing commands—not designing the underlying control policy—it is still important to understand the wide range of policies, as they may have differing effects on how swarms respond to human-provided commands. The policies used by robots in the OFFSET program were primarily biological and control-theoretic in nature, but the particular details are beyond the scope of this paper.

2.3. Human-Swarm Interactions

Another framework of swarm control for addressing the communication problems caused by the contrasting need for a distributed swarm with the need for a single, central operator involves using leaders within the swarm to act as intermediaries. Leaders can aggregate information from surrounding swarm members and summarize the data before forwarding to the operator, and thus lower bandwidth requirements, or can take human input directly and influence, either explicitly or implicitly, their surrounding neighbors (Walker et al., 2014). Still other solutions aim to remove the problems posed by limited bandwidth or high latency entirely by placing the human within the swarm itself, typically using speech or gestures for control (Nagi et al., 2014; Pourmehr et al., 2013) or even EEG data (Mondada et al., 2016). So-called proximal interactions present some benefits, such as more immediate feedback for the human operator and more immersion and awareness of immediate surroundings, but with the drawback of limiting the hardware available to the operator to what can be carried, such as a tablet computer (Divband Soorati et al., 2021).

While not developed explicitly for swarms, the Playbook approach (Miller et al., 2005; Miller and Parasuraman, 2007) for human control of multiple autonomous agents finds useful application in human-swarm interaction, and served as inspiration for parts of I3's design. Playbook uses the same approach sports teams use for determining how to accomplish goals on the field, and provides delegation capabilities, play calling and tuning, as well as play execution management to human supervisors interacting with semiautonomous agents. Plays represent generalized patterns of accomplishing a task in the form of broad constraints and objectives with details to be filled in at execution time by reasonable defaults, delegation to automation, or operator specification. Playbook thereby provides an operator the ability to specify an entire course of action for the team at varying levels of detail depending on need, workload, and the requirements of a given mission. Anything not specified by the operator before execution is filled in by automation so long as it remains within the constraints of the play. Figure 1 demonstrates how a play titled "Monitor Target" breaks down into lower level actions and potential alternatives.

While I3 does not implement Playbook strictly, many of the lessons from early Playbook work informed the design of other projects undertaken by the authors of this paper in recent years (Walker et al., 2019), which in turn heavily influenced the development of I3. OFFSET, and I3 specifically, makes use of "tactics"—a more narrow version of plays—to provide the operator a framework by which to execute commands at varying levels of detail to direct the robots under their control, see Section 3 for more detail.

2.4. VR and Other Swarm Control Interfaces

To our knowledge, I3 is the first VR-based swarm control interface deployed in real world environments with large numbers of real platforms (>100). In (Jang et al., 2021), the authors take a similar approach to the one in I3 for swarm control, where the human acts as a virtual "giant," overseeing a swarm from above and using hand gestures for control (e.g., to draw a virtual wall

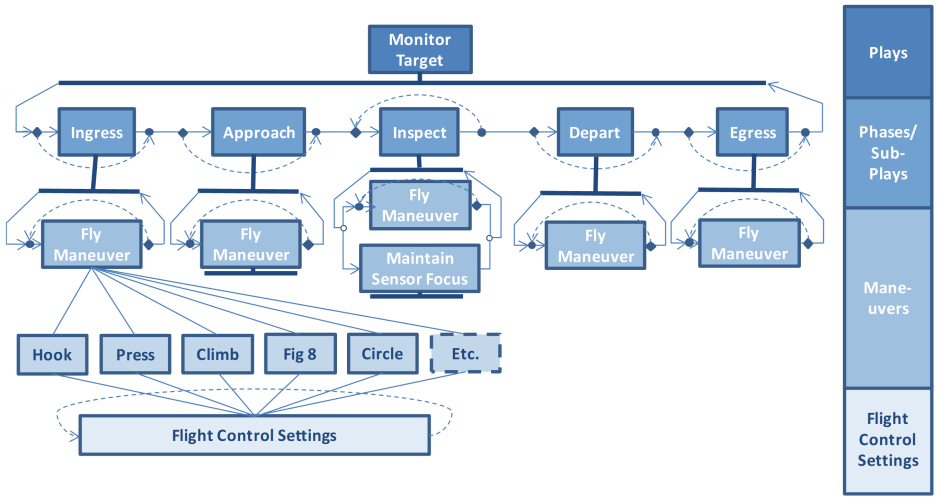


Figure 1. Definition of a `MonitorTarget` play illustrating the "space" of potential alternative methods of play accomplishment along with both hierarchical and sequential decomposition. After (Miller et al., 2013).

blocking movement across it). Our approach builds on this by increasing both the available actions one can take on the swarm, and by improving the visualization methods to show much larger swarm numbers over a larger area—eight city blocks instead of a single laboratory environment.

In (Haring et al., 2018), the authors demonstrate that a human can theoretically effectively control a swarm-like system in VR under manageable operator workload, however the authors there used a purely software, game-like environment to demonstrate their results. Other VR-based systems we reviewed similarly either use simulated agents or smaller numbers of real robots (<100) (Rajashekar et al., 2020; Weingart et al., 2018). Others have developed mixed reality interfaces for swarm control, with the goal of improving situational awareness by allowing the operator to be embedded in the real world while still receiving data from the swarm and imparting control (Das et al., 2017; Chen et al., 2020). Mixed reality provides some benefits over VR in terms of situational awareness, but is limited in the amount and control of the data presented in a way that virtual reality is not. These works are also limited in the number of robots being controlled, and the complexity of the tasks being performed.

In the future, I3 is intended to pair with other, complementary swarm interfaces such as tablets and other mixed reality interfaces used by humans in the field working alongside the swarm agents. One interface was demonstrated earlier on in the OFFSET program (Prabhakar et al., 2020). There, human operators could issue commands over a tablet interface and see the results in a separate virtual reality environment. We hope that future efforts in this field will be able to build off the results demonstrated by the OFFSET program, much in the same way OFFSET built on the results from the 2016 Service Academies Swarm Challenge (Chung et al., 2016).

The following sections will describe the system in detail (Section 3), how operators used I3 during the OFFSET scenarios and resulting metrics (Section 4), and describe the lessons learned and unique difficulties that real world VR-based swarm control entails (Section 5).

3. Immersive Interaction Interface (I3) Overview

Real world command and control over heterogenous swarms requires exploration of existing and new interface and control concepts. Instead of scaling linearly, as with traditional teleoperation and some multi agent control mechanisms, swarm command necessitates a control system in which a single operator can efficiently task hundreds of individual platforms while maintaining situational awareness (SA) of the environment. OFFSET focuses on urban operations, where the swarm conducts initial ISR tasking, eventually observing and controlling multiple city blocks.

Our Immersive Interaction Interface (I3) came about as a potential solution to this use case. Traditional interfaces, such as Tactical Situational Displays (TSDs), rely on a top-down map view annotated with entity and tasking symbology. This mechanism offers familiarity and ease of interaction with well known input modalities, but generally relies on representation of well defined entities. The OFFSET program challenges this traditional control system in multiple ways.

- Swarm groupings are a fluid concept, potentially representing a collection of mixed capability agents.
- Expressing verticality is critical for urban terrain, especially when considering multi-story buildings to observe and explore.
- Being able to reasonably express the volume of occupied space, including depth, is challenging for traditional control systems.
- It is desirable to support inspection of scenario elements from multiple perspectives, in terms of raw viewpoint as well as level of detail/abstraction.

Our response to these challenges takes the form of I3. I3 is a virtual reality interface built within the Unity game engine leveraging the capabilities of SteamVR and the Valve Index hardware system. The use of virtual reality places the swarm commander directly into the virtual battle space, enabling them to inspect and interact with their swarm at varying levels of detail and control in a manner similar to (Jang et al., 2021).

From the physical perspective, I3 is deployed "near the battle," a term we use to differentiate roles between multiple potential swarm interactions over both distance and time to engagement. The commander is connected to the swarm control network, but is positioned far enough back to support use of virtual reality hardware within a suitable physical environment. I3 receives live (or low latency) telemetry from platforms, while the user issues swarm commands in the form of tactics and mission plan engagements through the same communication medium.

When using I3, the operator wears a HMD (Head Mounted Display), which presents a three dimensional view of the scenario. Two controllers are used to inspect, interact with, and navigate within the world. An optional tracker strapped to the chest can enable separate reference frames for the head and body, allowing virtual side panels to appear over the shoulder of the I3 operator. The Valve Index system relies on outside-in VR tracking, so the hardware collection is rounded out with between two and four tripod-mounted tracking beacons. Attached to a capable laptop (which includes a high end video card), the I3 operator is thrust into the virtualized area of operations.

In this section, we will detail each of the mission tasks I3 is responsible for handling, and how we designed I3 to accomplish those tasks. In Section 5, we will discuss the difficulties encountered along the way, and where I3 could be improved in the future.

3.1. Visualizing the Environment

I3's virtual world is built on a sand table concept—a malleable workspace which can readily deform to create models of real locations. Unconstrained by physical and mechanical limitations, this space can support rapid perspective transitions, multimodal interaction, and unique visualization options unavailable elsewhere. Within this table space, the user transforms the world around them, both in terms of navigation and interaction with proxy elements to engage real-world behaviors.

The sand table space rests upon a hierarchy of transformations, permitting the user to manipulate the rotation, scale, and translation of the model while still maintaining spatial relationships between modeled elements. For scenario locations of OFFSET scale, it has been sufficient to treat coordinate translation as a mapping between Latitude, Longitude, and Altitude (mean sea level) into an XYZ reference frame defined in meters. Within this tree structure, we supply static world elements to define the environment in which we operate.

The most basic layer takes the form of a terrain mesh. We obtain publicly available elevation and aerial imagery, sometimes combining multiple resolutions. Through the use of external tools

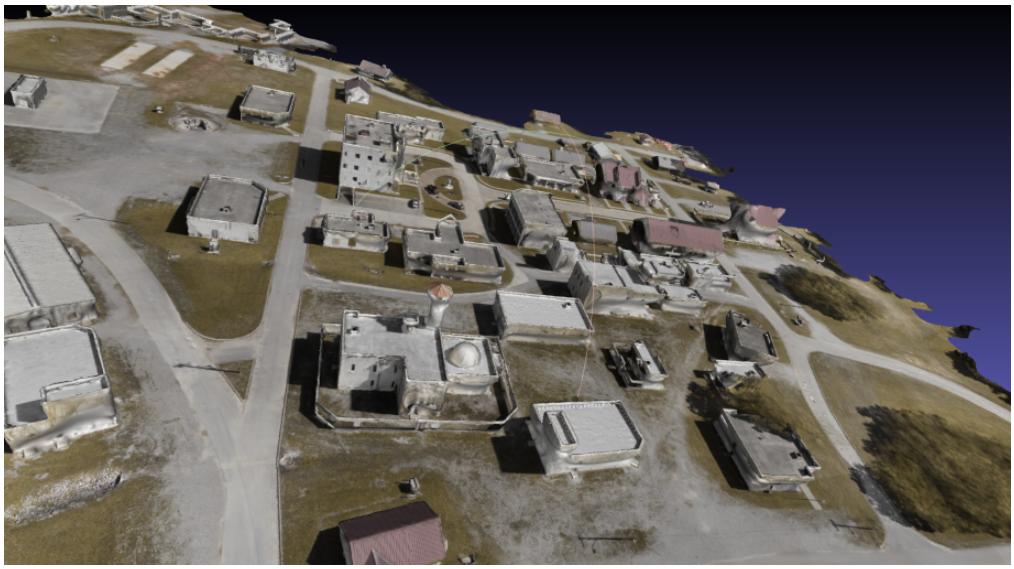


Figure 2. Detailed object model, which is decimated and then imported into I3 to be visualized in VR.

(e.g., GeoServer, GDAL) we extract rectangular elevation maps and images with concise geographic coordinate boundaries and resolution. This provides the initial terrain mesh for the scenario, which captures some elevation changes, but omits elements such as buildings, trees, and others. We usually leverage the digital terrain elevation data (DTED) or digital elevation model (DEM) to also provide a higher resolution elevation mapping service, since the coordinate space used by the rest of the CCAST system depends on above ground level (AGL) measurements.

The next layer comes from human-defined obstacle and building boundaries, in the form of keyhole markup language (KML) definitions. These are provided as part of the scenario intelligence, and in addition to assisting with simulation testing, permit I3 and the CCAST agents operating in the field to have a shared set of named geometry targets. Within I3, these are drawn as prisms, which we refer to as "extruded polygons."

The most significant input takes the form of an object model, generated by photogrammetry. This mesh includes anchor and coordinate space definition information, which I3 applies to affix it properly within the sand table space. Although computationally expensive, this addition provides unparalleled visualization of the location, frequently only a few days old by the time we begin trials (see Figure 2). On occasion, we supplement this world model with additional intelligence provided by external sources. For the final exercise (FX6), we could inject floor plans as geo-rectified images within our world model, enabling the operator to inspect the idealized interior of buildings while controlling the agents. Importing new models or KML files in I3 is easy, requiring only a change in the startup configuration file to point to the location of the files.

I3 supports visualization of a number of different world model elements and entities. While the presence of most are desirable to maintain SA and interaction hooks, we did identify conditions in which the operator found it useful to enable or disable entire classes of entities. We implemented toggles using the visualization panels options contained within the primary menu (see Section 3.5.1) for a number of features, including the following.

- **Buildings**. Named geometries capturing semantics of observable exteriors and potential interior exploration.
- **Obstacles**. Impassible regions used by platform route planners. Normally the swarm commander need not worry about the route planning obstacles, but on occasion it proved useful to visualize those bounds to better understand movement failures close to boundaries.

Figure 3. I3 operator using the right-hand controller in the x-ray mode to see inside a building.

- **Assets**. Individual platform models (e.g., UGVs and UAVs). Supporting hiding and showing the individual platform models became less important as we scaled up to tactic- and swarm-level visualization.
- **Mission Plan Elements**. Visualization elements of the scenario mission plan (Section 3.6). Our mission plan representation was fairly sensitive to the underlying structure's depth, breadth, and geographic proximity of task nodes. We often found it useful to suppress this visualization as we moved further into the scenario execution and relied on operator-directed tasking.

In addition to supporting load-time filters to manipulate the imported model's saturation and brightness, we implemented a customized, novel rendering shader to effectively support "x-ray" views inside of the object model (see Figure 3). Coupled with the ability to add simple floor plans and pose artifacts/platforms inside of these bounds, we could then better support interior operations. This feature could be toggled and off using the menu controls.

3.2. Navigation

Interaction with I3 occurs primarily through the use of the Valve Index controllers, which consist of a number of buttons, touch-sensitive surfaces, and analog track pads. I3 is sensitive to the position of the controllers within the world space, permitting us to identify accurate contexts for interactions—such as knowing the controller is within the bounds of a platform when a button is pressed. We use haptic feedback through the use of rumble units inside of the controllers to cue the user's attention to events of interest, but supplement this with visual elements to provide context. In addition to the controllers, I3 also captures the position and orientation of the HMD within the virtual space. As a consequence, the context system recognizes the central view axis.

Within I3, navigating the virtual space takes the form of shifting the world around the user. The left-hand controller is dedicated to these manipulations when the trigger is pressed, and supports scaling, rotation, and translation within reasonable min and max bounds. In addition, some throw momentum is included in the translation behavior, so that a rapid movement and release will result in continued translation slowly coming to a stop, allowing the user to quickly move themselves between far away positions.

These interaction mechanisms by no means eliminate the operator's ability to use the VR boundary space to physically walk through the virtual space, although the boundaries and physical

(a) 3DR Solo. **(b)** Uvify IFO-S.

(c) Modal AI VOXL m500. **(d)** AION R1 with TX2

Figure 4. Four of the platforms used by CCAST and controlled by I3 during scenario execution.

environment during the field events require us to discourage excessive physical stepping through that space, and rely on the controller-based world manipulation. Additionally, I3 supports scenario-defined sand table transformations, which effectively map into saved viewpoints. For larger scenarios of interest, we would record multiple saved views before mission execution to support quick transition between regions; however, moving the viewpoint outside the user's explicit, continuous control incurs some costs, so this was of limited utility. A continual transition to a new viewpoint can induce unease from the VR operator, while a transport/teleport (which is a favored mechanism on VR entertainment products) can cause some disorientation and loss of SA, forcing the operator to spend precious time regaining their bearings within the virtual world.

3.3. World Entities

Beyond the static world model, a number of entities are dynamically populated within the virtual space to represent both physical participants (agents, see Figure 4 for the platforms used during the OFFSET field exercises) as well as synthetic concepts (tactics). These are mapped into the sand table space using the previously-defined coordinate translation routines and maintain relative positions, size, and orientations. Each of the following entity types are capable of varied visualization depending on internal state, user interactions, as well as distance-based level of detail (LOD) capabilities native to the generic entity types.

3.3.1. Virtual Geometry

While many of the geographic reference points are either shared explicitly by coordinates with the agents or pre-seeded in the form of building or obstacle definitions, I3 also supports dynamic geometry creation. Within I3, the operator may specify a point, a polyline, a polygon, or an extruded (3D) polygon. The designation is performed using the right-hand controller to lay down discrete vertices and optionally defining a depth (in the case of the extruded geometries). The resulting geometries are available to the I3 operator as tactic parameters—such as a polygon to define the area of interest for a surveillance tactic, a polyline to describe waypoints in a route, and so on.

On application load, I3 parses the CCAST region file, mapping the KML entities into corresponding I3 geometries. We make a distinction between buildings and obstacles—the former are modeled from actual physical buildings identified within the scenario such that when the I3 operator issues an

`Explore` tactic for Building 17, as an example, the agent assigned to the tactic has full understanding of the building bounds and potential ingress points. Under some situations the I3 operator may find it useful to visualize nonbuilding obstacles for path planning purposes, but for the most part those geometries are ignored.

These shapes inhabit the same space as the detailed mesh model, allowing I3 to draw associations between spaces within the mesh and building identifiers, which is how the context system can operate when the mesh alone lacks those associations. Additionally, customized coloring is applied to buildings depending on scenario intelligence reporting suspected building contents.

3.3.2. Artifacts

As part of the test and evaluation infrastructure, OFFSET implemented proxies for real world entities using April tags (Olson, 2011). Easily identifiable by the platform's on-board image analysis tools, these tags represented elements ranging from general navigation hints (building identifiers, entry/exit markers), noncombatants, hostiles, coded intelligence, and high-value targets. I3 parses recognition events from the platforms and, using a lookup table, maps them into virtual entities called artifacts. Those artifacts are added to the sand table with customized visualization, allowing the operator to interact with them.

For instance, when a platform's camera recognizes an April tag representing a hostile red force element, the tag ID along with pose estimate is sent to I3. I3 then maps those coordinates into the virtual world space, loads the appropriate icon, adds a type-specific threat ring representing the known range at which the hostile can interact with platforms under operator control, and places it into the virtual environment. The operator can clearly see the advertised threat ranges, and if desired, task platforms to interact with the entity (to secure or suppress, in the scenario terminology). Using the LOD system, certain tags are hidden from operator view entirely (navigation hints), only shown when the operator's HMD is very close (noncombatants), or shown at all distances (hostiles and other hazards).

3.3.3. Agents

I3 represents individual platforms in the world space with a generic object model corresponding to their category (quadcopter, rover, or fixed wing). These models are intentionally made small and unobtrusive relative to the overall scenario scale, as we generally favor group tasking and status over that of individual platforms. These models can represent both real and simulated, live or replayed platforms.

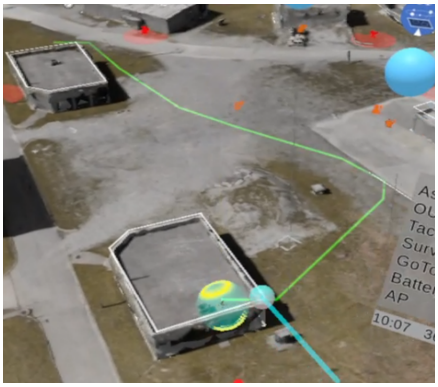
In order to facilitate some lower level interactions and inspection of vehicle state as the need arose over the course of the program, we implemented small status markers which color code according to the attached vehicle (see Figure 5a). These indicate whether I3 is receiving up-to-date telemetry from the platform, general tasking status, and whether the platform is currently struggling to deconflict a path to the target location. Additionally, the model itself changes color to indicate if the platform has been neutralized by scenario elements. Additional information from the platform can be queried using the entity inspection mechanism as detailed in Section 3.4, for example, the vehicle's current goal route (see Figure 5b).

3.3.4. Swarms

Much of the OFFSET interaction occurs at the swarm level, although the definition of "swarm" within the CCAST architecture differs slightly from other references in past literature. Within the current CCAST architecture, a swarm is defined as a shared label applied to platforms. These groupings can be defined by client software, enabling users to identify desired groupings for continual tasking.

Within I3, we synthesize swarm entities based on shared high level tasking—both to mission-plan based tactics and explicit operator tactics. For example, all platforms performing a cordon around a specific building (see Figure 6) are designated with the same swarm label, providing a custom-written volumetric shader to describe the space they occupy, and a reference handle to manipulate the tactic.

(a) Close-up of an individual UGV in I3. The lightning bolt indicates an electronic warfare payload, and the yellow ball representing current status (yellow denotes out of communication).



(b) Planned UAV route shown on controller hover.

Figure 5. Vehicle-specific data displayed within the I3 virtual sand table.

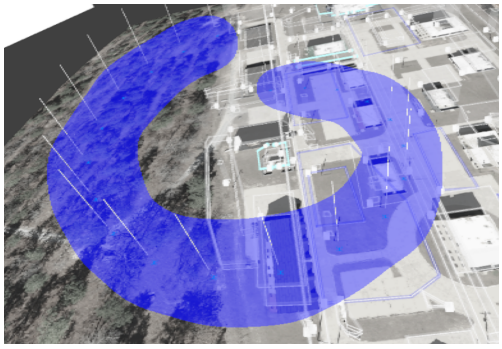


Figure 6. Visualization showing a swarm performing a `Cordon` tactic around a building.

Two visualization mechanisms are supported to describe these volumes. The first is a combination of ray tracing and signed distance functions (SDFs). The second approach applies kernel density estimates (KDE) to represent the density of the volume cloud—effectively, the density of the swarm. This visualization is a novel method for viewing swarm in real time in VR, although the implementation details are beyond the scope of this paper.

3.4. Entity Inspection

In addition to the visualization elements, the operator can directly inspect each entity (swarm, platform, hazard, etc.). Using the context-aware system, the right-hand controller can recognize when a cursor intersects with these entities and construct a summarizing glyph.

Figure 7 details the elements of the platform glyph, including platform type, payload, remaining battery, communication strength to the central dispatcher, current executing tactic, and whether the platform is real or simulated. On hover, I3 includes additional visualization elements for certain classes of entity. For platforms, I3 shows the current advertised route; for hazards (hostile artifacts), I3 shows a line to the tasked platforms; and for tactic visualizations, I3 highlights the associated platforms or swarms.

Furthermore, a summary panel, called the "clipboard" is shown above the controller model any time the user intersects the vehicle model with their controller cursor. This display shows summary

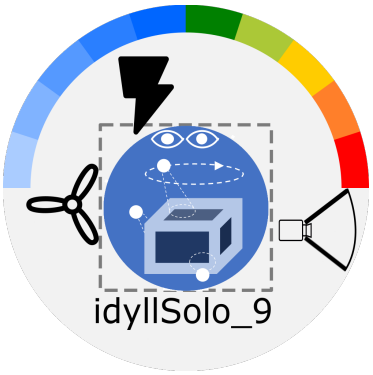


Figure 7. Example platform glyph. The colored bars at the top show communication connectivity (blue bars) and battery level (multi-colored bars). The lightning bolt indicates the platform has an electronic warfare payload, the propeller indicates it is a UAV, the camera indicates a forward facing camera onboard, the central icon indicates which tactic the platform is currently executing, and the gray dashed box indicates the platform is simulated.

information as text, to supplement the glyph icon, including asset callsign, current list of executing tactics, and fuel level.

Together, the glyph and clipboard help the user handle both neglect tolerance and neglect benevolence by providing insight into the current operations of the vehicle being inspected, and what will be executed (if anything) when the current tactic completes. By seeing the ongoing queue of tactics for a vehicle, the user can determine whether the vehicle can be ignored for longer or needs new tasking. Similarly for a swarm, the glyph will display the ongoing swarm-level tactic, and the individual vehicles' information displays will show the current subtactics, allowing a user to determine if the swarm is still in the early stages of a tactic where it should not be disturbed (e.g., the vehicles are still taking off), and when changes in the requested tactic are more beneficial (platforms are in-flight or returning to launch).

3.5. Menu System

We have tailored I3's design to suit the strengths of VR—favoring immersion within the 3D world to improve situational awareness and avoiding static panels and desktop-style menus. However, some situations, such as specifying details of a planned tactic, require a menu to ensure the operator is able to give precise input. In designing the menu system for I3, we aimed to provide this functionality while still maintaining immersion within the virtual world. Therefore the menu system is anchored around the controller location, in world space, where the user initiated the interaction, providing facilities to interact at the world level (visualization toggles, tactic menu, etc.) as well as context-sensitive queries or tactics.

Menus can be nested arbitrarily deep, can contain custom icons and visualizations, support multiple widget types, and for explicit buttons, can support both long and short click behaviors. There are two menu trees available to the I3 operator—the primary menu and the context menu, described in the following sections.

3.5.1. Primary Menu

The primary menu (see Figure 8) provides a static and predictable method to execute the majority of available actions within I3—visualization toggles, the geometry creation menu, tactic menu, x-ray controls, and mission plan controls. Below is a list of the options and submenus within the primary menu, along with their functionality.

- **Toggles**. This control set supports the operator in explicitly enabling and disabling certain entity visualization classes—artifacts, buildings, platforms, swarms, world model elements, etc.

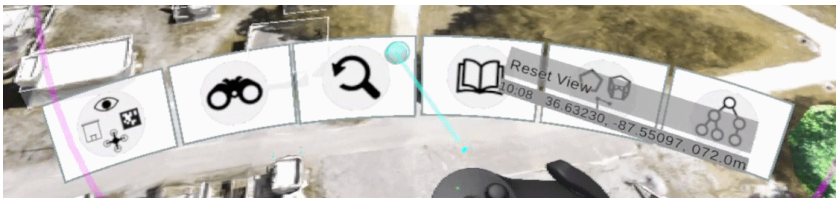


Figure 8. Top level menu controls.

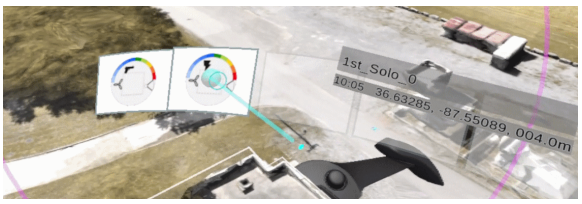


Figure 9. Operator using the context system to quickly select between two close platforms.

- **X-Ray**. I3 supports two distinct x-ray modes—the first applies a cylindrical exclusion zone extending from the user's viewpoint to the cursor location, while the second creates a spherical zone expanding from the cursor (see Section 3.1).
- **Views**. Within this submenu, the user may reset the view to the initial state or quickly switch to another preset view, as described in Section 3.2.
- **Tactics**. This menu provides a method of selecting and fully specifying any of the supported tactics to be issued to CCAST platforms.
- **Geometry**. Within this menu system, the user may define new points, polylines, polygons, or extruded polygons, which may then be used as targets for tactics, as described in Section 3.3.1.
- **Mission Plan**. This submenu contains controls to fetch, upload, and instantiate mission plans (see Section 3.6).

3.5.2. Context Menu

The context menu compliments the primary menu by offering the capability to query or engage behaviors based upon what is in proximity of the controller (see Figure 9). This behavior is supported by most, but not all, of the various entities—namely: buildings, artifacts, platforms, swarms, tactic visualization nodes, and mission plan elements.

The initial row of the context menu is populated by references to all entities close to the operator's cursor. If there are multiple, they are sorted by distance to the interaction point up to a certain maximum threshold. This supports both quick selection within a sparse location, but also enables the tactician to interact in a dense location, then picking out the specific artifact they wish to engage.

The typical usage of the context menu is to speed up tactic invocation when the operator does not wish to formally specify all of the tactic details, instead leaving many to their default value. A common example is interacting with an in-air UAV and tasking it to return to the launch pad immediately. The operator also has the capability to interact with a building element and request an immediate `SurveilObject` tactic—permitting the dispatcher to auto-allocate suitable platforms—or transition into the tactic calling menu while pre-supplying the designated building as the target of the tactic. Where possible, we design for and exercise this level of interaction during the field events.

3.6. Mission Plans

I3 relies on XML-defined mission plans to establish initial tasking for scenario execution. Within the CCAST architecture, mission plans represent a series of tactic "nodes", each containing one or

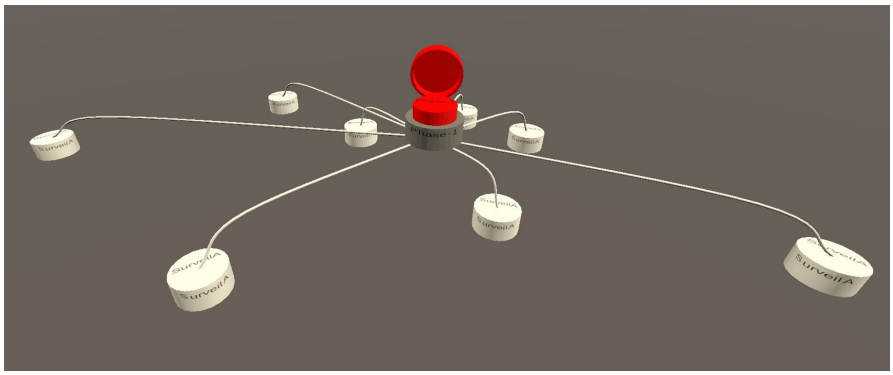


Figure 10. Standalone mission plan visualization, showing several nodes (white disks) gated by a single signal (red button with raised cover).

Figure 11. HUD present at the top of the I3 operator's view throughout execution.

more bound tactics. Tactics may begin at mission start, rely on explicit signals issued by operators or software, or execute upon completion conditions asserted on predecessor tactics.

Within I3, we visualize these structures above the sand table as a hierarchical tree (see Figure 10)—first defined by top level signals, and subsequent levels conforming to the tactic completion dependencies. The physical positions of the signal and tactic nodes are generated from the centroid of associated tactic geometries, then deconflicted using repulsion physics in a similar manner to force-directed graphs.

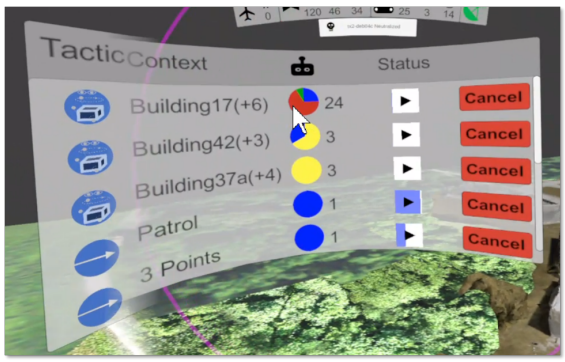
Operator control over the mission plan involves triggering of various signals. These signals gate execution of one or more mission plan nodes. Through this mechanism, we can move through the scenario phases as the user determines conditions are suitable. For instance, most fielded missions plans involve an initial series of surveillance tactics. These are frequently deconflicted by region to reduce the risk of in-air collisions during UAV transits to or from the staging area. Each region is identified by a different discrete signal.

The operator can engage signals by interacting with the associated nodes. When the operator hovers their controller over a specific mission plan node, the platforms and geometries associated with the specific subtactics of that node highlight within the sand table.

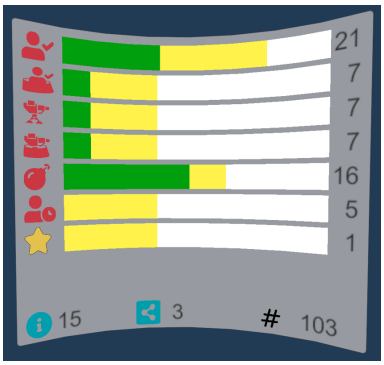
3.7. Information Readouts

While the majority of information available to the user is embedded within the sand table representation in the form of asset positions, icons, other interactable entities, summary information about the scenario as a whole must be presented in a centralized location. To that end, we developed two types of displays: an overhead heads-up display (HUD) and summary side panels.

The HUD presents always-available summary information related to the currently deployed assets (Figure 11) and connectivity between the I3 operator and the swarm. Included on the HUD is an indicator to supply current telemetry status—useful for diagnosing communication issues—as well as constantly updated tallies of platforms counts by type and class. Lastly, a notification pane displays critical information about events as they occur in the world, including new scenario intelligence sightings or platform neutralizations. When no notifications are available, this pane is hidden.

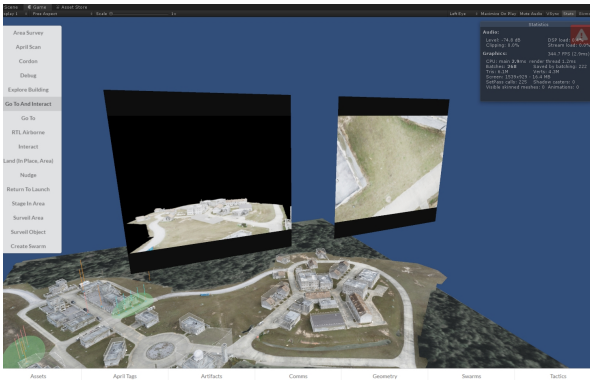


(a) I3 tactics panel, showing ongoing tactics issued by the operator or part of the top level mission plan.



(b) I3 hazard summary panel, showing counts of all hazard categories encountered so far in the scenario.

Figure 12. Information panels displayed on the left (tactics panel) and right (hazards panel) sides of the operator when the VR chest tracker is present.



(a) Updated mjpeg stream published from both simulated UGVs (shown here) and live UGVs.

(b) Video feed streaming from a fixed wing platform during live flight.

Figure 13. Video feed windows used within I3 during the OFFSET program.

The operator's inertial frame, established by using an additional tracker, proved a useful attachment point for side panels. These information readouts are always available at a glance to the operator's side (see Figure 12). The tasking panel, on the operator's left-hand side, provides an enumeration of top level tactics being executed within the scenario, supplying a summary of the tactic type, target, composition, and state. Interaction with the row supplies a convenient handle to terminate the tactic. The hazard plane, on the operator's right-hand side, presents scenario-specific hazard and item-of-interest listings, emphasizing status of threats, including segregation into classes indicating tasking status.

3.8. Platform Video Streaming

While the data feed to I3 for platform information is aggregated at the CCAST central dispatcher, each platform has one or more discrete cameras, which with the correct extensions can support video streaming. Figure 13 shows how video feeds appear in I3 for both simulated and real platforms. I3 can display multiple video streams, and they can be anchored on the operator's controller, or alternatively anchored over the platform itself, to move alongside it as the platform traverses the environment.

4. I3 Usage During OFFSET Exercises

Each successive OFFSET field event included expansion of swarm size, capabilities, environment bounds, and autonomy, requiring significant expansion in I3 functionality and the grounding of the interface design to support scenario-specific requirements and the complications of real world operation. Both swarm level control concepts and practicalities associated with field testing of the complex autonomy stack influenced the choices made during I3's development. The work presented in the previous section is the end result of this iterative development, and in this section we describe how I3 was using *during* a single shift, of which there were usually 10–20 during any given exercise.

4.1. Preparations

Each field exercise begins with a series of activities to populate the I3 model and behavior sets. We use a GeoServer to create the initial terrain mesh, and observations from publicly-available maps define the initial geometries of interest—both keep out zones and buildings for tactic targets. The test and evaluation team define scenario activities and requirements, which inform development of new tactics and visualization or interaction elements necessary within I3.

Next, we conduct a site survey, which generates the complex object model (see Section 3.1) used as the basis for the I3 world model. We also refine building and obstacle bounds, ideal LTE base station points, and the desired network topology alongside this effort, which in turn influence the position of the primary command center where I3 is located.

As the understanding of platform capabilities solidifies during the exercise, modifications to CCAST execution are often necessary to better meet scenario requirements. This often takes the place of tactic modifications—both downselecting appropriate tactics (and their templates), as well as designing new composite tactics from basic primitives available as part of the existing tactic library. Occasionally, this requires modifications to existing visualization components, such as enabling flags for certain platform status elements or including emphasis of sand table entities given intelligence found during scenario runs.

Ideally before missions, we perform trial runs of proposed mission plans against generic red-force positions. The simulation provides idealized conditions in which to measure our strategic engagement of the scenario. While this frequently downplays realistic deficiencies we meet in the field, it enables us to exercise the initial mission plan and interaction concepts.

4.1.1. Shift Briefing

In the field, each exercise shift is preceded with a mission briefing (see Figure 14). The CCAST team defines an operational plan to approach the scenario—swarm composition by the platform type and capability, staging and recovery areas, locations of interest, and initial tactic-based mission plan.

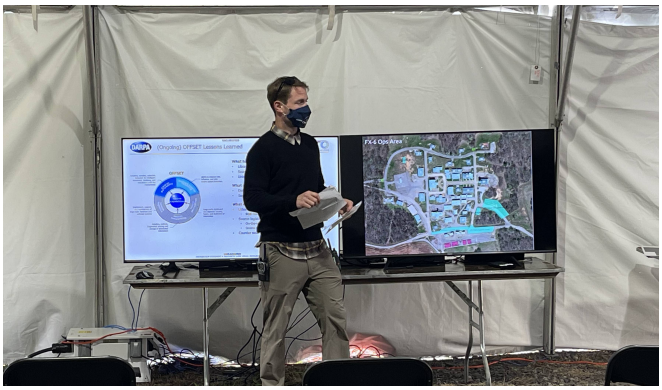


Figure 14. Mission briefing by the CCAST team lead.

Figure 15. I3 setup in command and control room.

This briefing includes the entire team, as well as safety spotters and evaluators, ensuring all have proper understanding of anticipated actions and expectations of performance.

4.1.2. Physical Setup

The ideal environment for I3 is a protected from the elements and stable, although we frequently find ourselves in less than ideal conditions. In two of the field exercises, we controlled the swarm from an open pavilion, offering an impressive vista of the platform staging area, but leaving the VR system vulnerable to gusts of wind, which shake the tracking beacons. In other cases, we deployed I3 in cinder block rooms, which suffered from poor acoustics and frequent foot traffic through the work space. Many of the events were held in environments outside the operating range of the VR hardware—below freezing on one end of the spectrum, high heat and humidity on the other. While these were less than ideal, I3 works in a surprising range of environments, including the back of a minivan, powering the VR hardware through DC inverters. We expect such setups to become more robust and practical as the technology improves.

The I3 space (see Figure 15) generally is occupied by both the I3 swarm commander as well as a second team member running data collection. Frequently, additional team members liaison with the technical team to coordinate safe flights, range clearance, and tactical consideration. Throughout the exercises, we placed emphasis on predictable and safe operation of the swarm platforms above comfort or the ideal operating conditions of equipment.

4.2. Mission Execution

With all participants in place, the I3 operator observes the swarm composition growing as additional platforms are brought online. During this period, the engineering team performs diagnostic tests, and will on occasion instruct the safety team to remove physical platforms if they do not pass initial checks. Once the team observes the active platform count is sufficient to meet the mission objectives, operations are cleared with the range safety officer, and execution begins.

Throughout the scenario, takeoff events are loudly broadcast over the safety channel to ensure all spotters and observers understand the anticipated movement. I3 receives live telemetry from the participants, and the operator can inspect agents, artifacts, swarms, and other entities present in the virtual sand table space. The operator monitors the execution of tactics, both within and outside of the mission plan context, artifact, and hazard counts, and engages high level tactics to further the scenario play.

4.2.1. Mission Plan Engagements

The initial portion of the scenario revolves around the mission plan structure. As discussed in Section 3.6, the mission plan describes a collection of sequential tactic groups to execute. In some

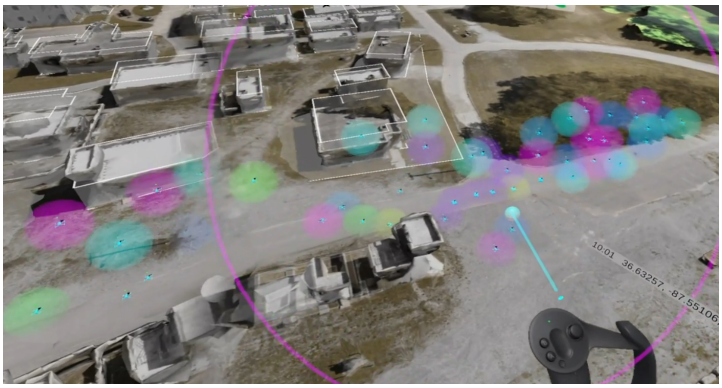


Figure 16. Initial volley of swarm surveillance tactics at the start of one mission. The magenta ring represents the extent of the operator view within VR.

cases, the operator controls the mission plan through explicit signals, while in others, completion criteria (e.g., success or failure) from predecessor plan elements trigger further tactics.

I3 begins by pulling a mission plan from the dispatcher service. This mission plan evolves over the course of a field exercise, growing in complexity and capability as tactics and operational plans are refined through multiple shifts. The I3 operator can view the geographic associations, and if necessary, instruct the dispatcher to allocate platforms to the tactics based upon capability requirements. When all participants are prepared, the I3 operator issues the mission start signal.

Typically, a mission plan begins with a number of surveillance volleys—sending groups of UAVs out to scan both buildings and areas (see Figure 16). At the same time, ground-based tactics from the plan instruct UGVs to being patrols to identify artifacts throughout the environment. These are automated behaviors, requiring no human interaction beyond the issued signal. The operator acts as a conductor, providing timing for high level triggers, while observing the plan execution to ensure it meets their requirements. Some platforms are neutralized by hazard artifacts and automatically attempt to move to an identified medic location to revive and re-enter standard operation. Some platforms fail to start and some UAVs inevitably crash, which are documented for recovery and diagnosis after scenario completion.

Throughout this process, intel reports stream in to I3, identifying and localizing artifacts of interest within the scenario. The tactical picture comes into view, and much of the mission plan tasking is quickly exhausted.

4.2.2. Tactical Engagements

The latter portions of a shift tend to focus on selective, lower-level engagements. The operator can and still performs some high level tasking (for example, to conduct surveillance on a given cluster of buildings), but much of the focus shifts more to individual scenario elements, including the following.

- **Surveillance**. The operator identifies gaps in the initial coverage or desires a closer look at a ground region. They can designate a polygon (or select a building) and instruct the automation to identify platforms to scan for artifacts (see Figure 17).
- **Patrols**. Groups of one or more ground platforms can be configured to patrol for threats or to reposition the rovers within another staging area in preparation for later actions.
- **Breach**. The operator can request certain platforms with matching capabilities to explore buildings, either explicitly identifying the agent or relying on the allocation routines to find a suitable candidate.
- **Interaction**. As hazard artifacts appear and are further localized, the operator can instruct platforms to engage, either by explicitly selecting them or by allowing the dispatcher to allocate

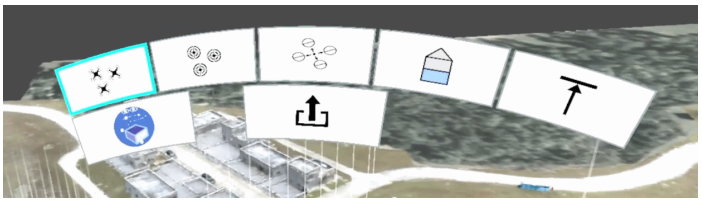


Figure 17. Operator navigating a tactic-specific (`SurveilObject`) submenu to supply desired parameters.

Figure 18. Two swarm commanders operating concurrently using two separate instances of I3.

automatically. Some effects are permanent, and some are active only while a platform remains in proximity.

Whenever possible, the operator skips per-platform interaction, and instead relies on the allocation routines. The context menu (see Section 3.5.2) aids in this task, supporting quick interactions to state the goal behavior, while the actual assignment of swarm members to perform the tactic is left to a dispatcher algorithm weighing capabilities, current tasking, and proximity. However, there are times at which the operator must manually task platforms, sometimes at a waypoint-by-waypoint basis, and although disincentivized, I3 supports this mode of interaction when required.

4.3. Multi-User Experience

To support the program's swarm commander role, the CCAST system needs to enable simultaneous control over hundreds of heterogeneous physical platforms. During the final field exercise, we attempted a single trial using *two* operators in different instances of I3 for the first time (see Figure 18). Nothing within the architecture, I3 or CCAST, prevents multiple sources from issues commands concurrently and consuming telemetry. On the other hand, there are no mechanisms for partitioning such control, either at the platform resource level or tasking level. We had previously provided the capability for multiple operators to be connected to the swarm, but in previous runs explicitly forbade the secondary instance from engaging tactics, instead placing their instance of I3 into a "read-only" mode.

During this final run with two operators, we colocated both operators in the same command room. Tactic execution was differentiated by callsign, such that one operator could observe a tactic visualization within the world space and identify who instantiated it (themselves or the other commander). Because the system inherently had no restrictions or mechanisms in place to enforce responsibilities and resources to each operator, we elected to deconflict geographically. For explicit tasking, each operator assumed control of half of the area of operations. During the experience, the operators called out to each other before tasking near the fictional boundary line to minimize surprise and risk of conflicted use.

While not a thorough vetting of the concept, this experience informed some necessary design considerations for future efforts:

- **Responsibilities**. We focused on geographic deconfliction when dealing with tactical engagements. Other potential approaches are differentiating by domain (air or ground), or even engagement level (mission plan, surveillance tasks, building search). Each potential organization would benefit from interface customizations emphasizing those interaction components.
- **Allocation**. While it may be desirable to consider the entire swarm as the asset pool for automated allocation routines, it may be useful to consider some segregation (or pre-allocation) of the per-operator resources. The I3 operator has awareness of idle agents to support their current tasking, but in a shared environment those resources can be quickly assigned elsewhere.
- **Goal Tasking**. The standard I3 workflow involves creating tactics to satisfy an operator goal or objective. The actual context of the goal is not expressed within the tactic, so another operator may only try to infer what the original caller intended to accomplish. A more expressive command structure may enable the first operator to convey a series of discrete tactics, the sum of which readily reflects the intended effect. With such a system, the second operator could effectively take over management of the original goal, without needing to communicate in depth with the other operator.

4.4. Exercise Metrics and Results

The nature of the OFFSET program makes it difficult to compare I3 to existing methods and determine how well (or not) I3 is performing at a given task, primarily because the task is, to our knowledge, completely novel in the literature. Never before has a single operator attempted to control a swarm of hundreds of real heterogeneous platforms distributed over multiple city blocks performing a wide variety of tasks—from surveillance to building exploration, close quarters interaction, and coordinated engagement with identified targets. The fact that I3 was able to give a single operator the ability to accomplish even parts of the scenario already represents progress over the current state-of-the-art. However, it is useful to investigate how well I3 accomplished the stated goals of the program, as well as how I3 improved at this task over the course of multiple field exercises.

The ultimate goal of the OFFSET program was to control up to 250 assets over eight city blocks performing generalized ISR and interaction with identified targets of interest. By that metric, I3 was a success: the CCAST system, with a single commander at I3, controlled up to 226 assets during the final field event, and successfully performed surveillance and identification of various threats and hazards on every building in play. Figure 19 shows the total number of assets under I3 control across the shifts during the final three field exercises of the OFFSET program.

The total number of assets under operator control only tells a small part of the story, however. More important when evaluating I3 is how many assets were *actively tasked* throughout a shift. Over the course of the program, we made significant improvements to the interface designed to cut down on visual clutter and allow faster tasking of assets. Figure 20 shows the number of nonidle (tasked) assets on average each shift throughout the final three field exercises. Although the earlier shifts are similar across the field events, the longer length of the final exercise (FX6) allowed us to better refine our procedures and improve I3 further to enable much greater numbers that before. Results show that the I3 operators were able to task more assets during FX6 than both FX3 ($t = 98.637$, $p < 0.001$) and FX4 ($t = 4.104$, $p < 0.001$).

Furthermore, if we investigate the share of tasked assets across time *within* a shift, we find that by the end of the program, during FX6, the I3 operator was able to begin tasking higher numbers of assets much more quickly than before (see Figure 21).

The most likely explanation is that between FX4 and FX6 we made significant improvements to both the mission plan interface and the context menu, both of which provide ways to get assets performing a task much faster than the original menu-based tasking system. The level-of-detail

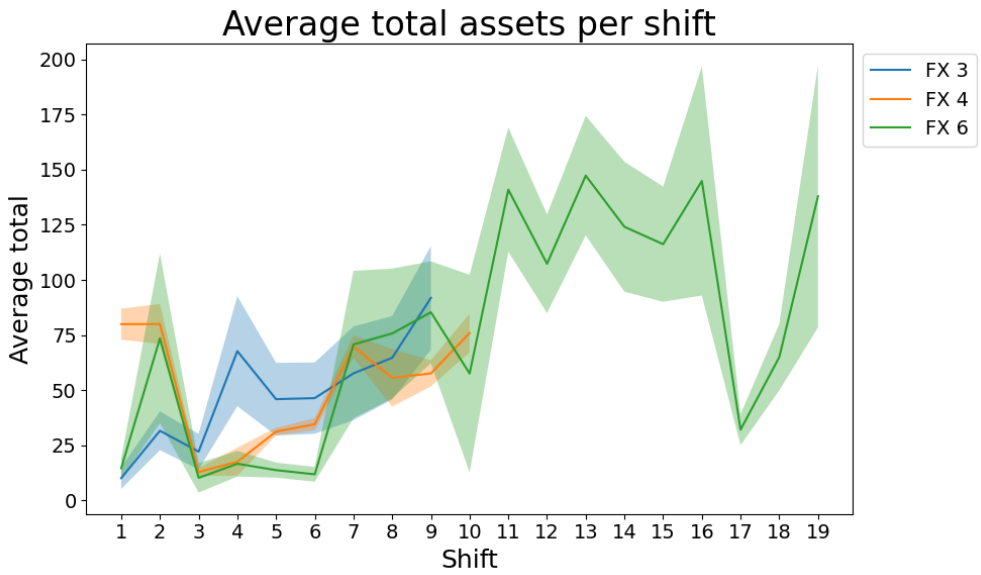


Figure 19. Average number of assets under I3 control across shifts at each flight exercise (FX). Note that FX5 was cancelled due to the COVID-19 pandemic, and that FX6 lasted considerably longer than previous events. The shaded regions represent standard deviation around the average. Shifts are ordered chronologically.

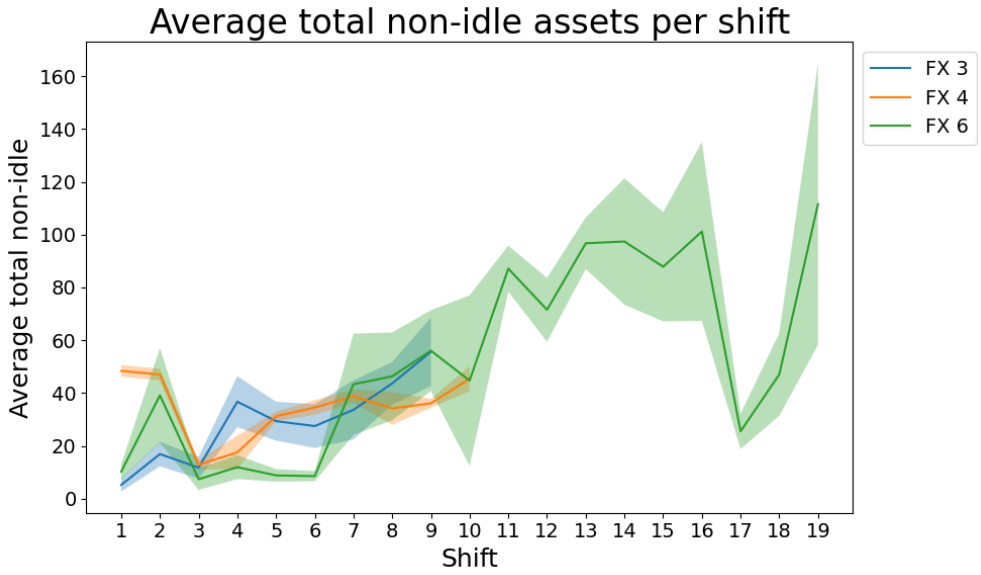


Figure 20. Average number of nonidle (tasked) assets under I3 control across shifts at each flight exercise (FX). The shaded regions represent standard deviation around the average. Shifts are ordered chronologically.

(LOD) system also reduced visual clutter, and along with the HUD, which provided counts of tasked versus idle assets, likely assisted the operator in picking out idle assets in the virtual environment. Investigating tactics issued across each of the flight exercises supports these findings. Due to how data was gathered, we don't have individual tactic data for FX3, but comparing FX4 and FX6 shows that significantly more tactics were issued during the latter (147 tactics per shift during FX6 versus 33 per shift during FX4).

It should be noted that there exist some limitations to the conclusions we can draw from these findings. First, there were moments during the exercises where operation had to pause, due to

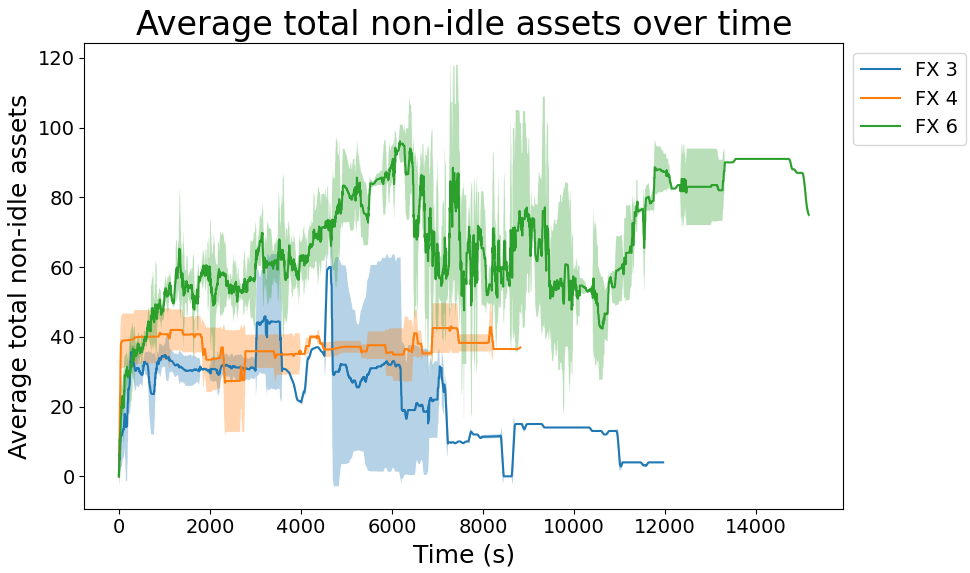


Figure 21. Average number of nonidle (tasked) assets under I3 control across time within shifts at each flight exercise (FX). The shaded regions represent standard deviation around the average. Shifts are ordered chronologically.

safety, weather, or logistical concerns, and these moments are not easy to separate out from the "active" scenario moments in the data. Second, there are always communications or hardware failures inherent in field robotics, especially with a swarm of this size, so many times idle assets were not truly idle, but rather untaskable due to some failure unknown to the I3 operator. Finally, in some cases leaving assets idle is desirable, as you may want to keep some in reserve pending the result of a currently executing tactic. Despite these caveats, the relative distribution of pauses, failures, and other unforeseen circumstances remained uniform throughout the exercises, making trends still useful in determining whether I3 was improving as an interface to the CCAST system.

5. Discussion

The OFFSET program aimed to move away from reliance on single-vehicle tasking and teleoperation and towards high-level swarm tasking. Therefore we designed I3 from the beginning with this goal in mind. Especially early in the scenarios, I3 achieved this goal. Tactics like `SurveilObject` and `SurveyArea`, along with the predefined mission plans fetched from the central CCAST dispatcher, served as the main drivers of vehicle behavior during initial stages of the shifts, allowing an operator to reliably task large numbers of vehicles with few actions in VR. Outside of these initial stages, I3 supported high-level interaction through carefully designed interfaces. As an example, the I3 context menu allowed an operator to call a building surveil by five platforms—one with a downward-facing camera for the roof, and four with forward-facing cameras for the sides—in as few as two clicks. One click specified the target building, and another to issue the tactic request to the dispatcher. Much of the time immediately following these actions focused on monitoring the evolution of these high-level tactics and preparing for what comes next, which is precisely how I3 and the OFFSET program in general were designed.

However, later stages of the mission often devolved into more low-level, individual tasking (Figure 22), especially as automation and communication failures between I3 and the robots mounted. Additionally, later stages of the scenario called for individual, point-based interactions between vehicles and scenario elements (e.g., to neutralize a hazard in a doorway), which requires either individual operator attention, or a high degree of automation on board the platforms or dispatcher.

Figure 22. I3 operator selecting an individual platform of the swarm for assignment to a new tactic.

The friction between high- and low-level tasking did not come as a surprise, however, as OFFSET is a program designed to push the boundaries of both swarm hardware and communication infrastructure as well as interface design and control structure. Because the interface can only be as effective as the underlying automation and sensor data, and in turn the automation relies on well-defined high level tasking through the interface, a failure on one side often led to a failure on the other. In our daily lives, we routinely interact with automation refined and perfected over decades of testing, and the associated failures involved, without thinking about how this automation depends on both reliable human input and correct operation of other automation present in the system. Therefore it is not surprising that we see such difficulty when using I3, especially considering that it is an attempt to advance the state of the art. The following subsections will detail some of the missteps and lessons learned throughout the development of I3 and where this work could be improved in future research and development.

5.1. Visualizing the Environment

As pictured in Figure 23, the initial concept for I3 revolved around a control room. There would be a central element within the virtual room representing the sand table visualization space. Around the room were to be various analogs to real world objects—cabinets you could interact with to pull out maps, large displays to mimic war-room like screens, and so on.

Early implementations included large billboard-like displays recessed within the extents of the room (see Figure 23). These were intended to provide classes of information to the operator—one designated to enumerate artifact recognition within the scenario, another listing the swarm composition, one providing a top-down map display, and so on. These displays were updated in real time as telemetry reached I3 from the fielded swarm.

Other than the core sand table concept, the control room features were removed over the course of the program as we gained additional experience during the numerous exercises. By originally maintaining the façade of the fixed table itself being a physically occupied space, I3 discouraged the user from making the most of the virtual elements. Early on we found that operators would lean over (and into) the table surface volume, awkwardly, while there were no real restrictions forcing them to maintain distance. This became particularly problematic when they desired to interact with elements across the table space. The control panel buttons were a useful attachment point for the manipulation of the table visualizations and transformation, but were difficult to engage if not

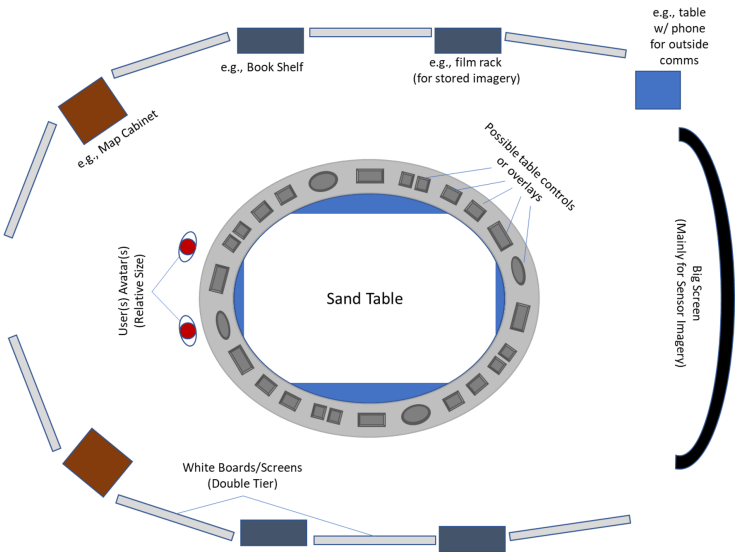


Figure 23. Initial concept design for the virtual I3 control room.

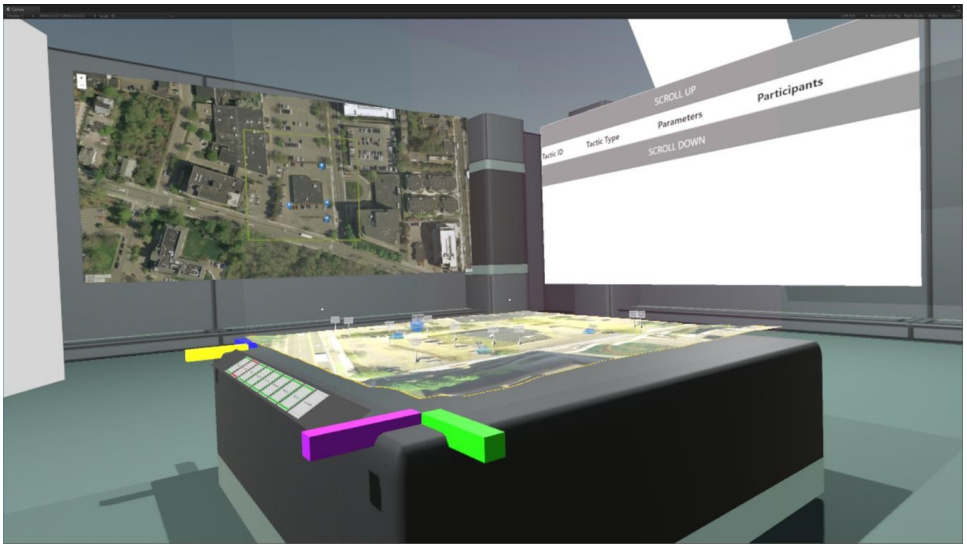


Figure 24. Initial implementation of the I3 sand table and control room. Billboards can be seen along the back wall, and the left side of the sand table contains buttons for manipulating the environment.

facing the table appropriately—depending on the operator staying on a specific side or orientation to the table.

While the original wall mounted billboards (see Figure 24) did provide critical information, their placement was less than ideal. Especially with earlier generation VR kits, effective resolution meant any text needed to be surprisingly large to be legible, especially at distance. Any measure of interaction with those displays required the use of a laser-pointer type device to click or engage on controls, the stability of which was vulnerable to small tracking errors or involuntary hand or arm movements. As with the control panel, their fixed nature in the room meant the operator needed to maintain awareness of their orientation in both the room space and sand table space to maintain awareness of their environment and to understand where to look for specific information, including potentially directly behind them. Therefore we removed the fixed billboards and moved

Figure 25. Initial wrist menu, projected above the operator's horizontal left arm, interacted with a pointer from the right controller. This offered some utility, but overall too limited and awkward to support our operations.

instead toward embedding as much information as possible into the virtual world on the sand table itself, or on to smaller displays attached to the operator's reference frame.

5.2. Menus

Similarly, our original menus existed at fixed locations within the sand table space, either on a distant display or offset from the user's reference frame. After multiple field exercises, we found a large amount of attention shifting during tactic calls and general interactions, where the user would need to explicitly glance back and forth between the area of interest on the sand table and the offset menu display. By adjusting the menu placement to an unobtrusive location relative to the controller as described in Section 3.5 and enabling the user to move the menu when needed, we found the experience less awkward.

In fact, the final state of the menu system evolved through multiple rounds of design and implementation (see Figure 25 for an earlier implementation). Throughout the program history, this interface grew out of a balancing act between different priorities or considerations listed below.

- **Speed**. Improving the efficiency of the interface by judging the speed at which we can interact with the menu—both in terms of discrete inputs (button presses, controller positions) and time to execute.
- **Extensibility**. Defining menuing templates which were easy to extend or instantiate to incorporate new controls or tactics became increasingly important as more tactics entered usage.
- **Coherence**. Shifting focus between a 2D display and the sand table world is costly, both mentally and physically.
- **Cost**. Development time being limited, the effort and time to design and evaluate a solution needs to be weighed against other feature requirements.
- **Practicality**. This final point specifically applies to the field evaluations conducted within the program. An approach to menuing or general system interaction, no matter how refined or well functioning in a lab, needs to be revised (or discarded) if it does not perform as needed within the field.

5.3. Platform Allocation and Tactic Feedback

5.3.1. Wildcard Tactics

Wherever possible, the CCAST dispatcher handled allocation of individual platforms in the field to the tactics requested by the I3 operator. Here, we see a clear influence of prior Playbook-related work:

I3 would send a request to the dispatcher to execute a certain tactic along with either fixed default parameters (e.g., stay between 10 and 50 meters above ground level, use up to five platforms, etc.), or overrides provided by the operator. Upon receiving the request, the dispatcher would survey the available platforms—those that had the capabilities required by the tactic, sufficient battery level, and not already tasked—and assign the most suitable ones to the new tactic. If such an allocation did not exist, or the command was ill-formed on arrival, the dispatcher would report as much back to I3. In cases where the operator desired more granular control, they could individually select the platforms to execute the tactic, and in some cases decompose the tactic hierarchy themselves and issue the lower level commands. For instance, the I3 operator could issue a `Patrol` tactic to a set of ground vehicles, which instructed them to move along a route, survey for hazards along the way, and disarm them when discovered. The operator could also instead issue individual waypoints and handle the interactions manually with the same or different platforms as new hazards were discovered and appeared in the interface.

Implementing support for these "wildcard" tactics, which allowed the I3 operator to rely on the dispatcher for much of the low-level allocation and parameter setting, significantly increased the number of actions and intermediate goals the operator could accomplish over the course of the mission. When contrasting exercises earlier in the program to later ones, the benefit became clear.

However, the realities of the environment, the current state-of-the-art in hardware and automation, and the changing requirements of the scenario often necessitated a lower level of control than was originally designed for, oftentimes to the detriment of operator situational awareness, time remaining to accomplish the mission, and workload. For instance, some platforms would experience silent hardware or communications malfunctions, especially as the scenario went on, in a way that was unknown to the dispatcher but clearly inferred by the I3 operator based off the individual platform's prior performance. Therefore the operator would not want to leave to chance the possibility that the dispatcher would select the faulty platforms during automatic allocation, and instead preferred to manually select known working ones.

5.3.2. Real versus Simulated Assets

Another point of difficulty came from the desire to stress test the number of assets I3 could handle, along with what the CCAST team could properly utilize, by introducing simulated vehicles alongside real ones. We designed I3 to be agnostic to whether a platform was real or simulated in order to realize this goal; however, the distinction became important when the dispatcher would select a mix of the two during allocation, and safety spotters on the ground were therefore unaware of how many platforms to expect in the air. Furthermore, certain scenario objectives required real platforms to accomplish, which meant the I3 operator needed to manually ensure real platforms participated in the tactics intended to accomplish them. Therefore we found a need to design a change to the standard asset visualization in order to easily distinguish between real and simulated assets.

5.3.3. Feedback on Tactic Progress

I3 sought to provide feedback about tactic progress and potential failures wherever possible to improve operator SA and allow the operator to quickly rectify the situation. When the dispatcher received a malformed tactic instruction, or could not find a valid allocation for a tactic, I3 received this information and would notify the operator with both auditory and visual signals (see Section 5.4)—a brief message stating the tactic name that failed and a change to the tactic's visualization in the sand table, respectively. This feedback proved invaluable as the scenarios went on and more failures occurred due to mounting hardware failures. However, tactics would occasionally fail silently and for unclear reasons, often requiring new or redesigned components within I3 to handle them. Examples include communication errors between I3 and the dispatcher (or between the dispatcher and the individual platforms), new hardware or software failures on the tasked platform, the platform being stuck in the terrain, and double tasking. The last of these phenomena occurs when a platform is already performing another tactic at a higher priority, and thus waiting to begin the newly issued tactic until the previous one completes. Over the course of the program,

we extended I3 to better handle this type of failure by adding the context-aware inspection, which provided the operator an easily accessible view into lower level platform states by hovering over the platform in the sand table. There, the operator could see the tactic queue of the platform and recognize double tasking.

Another improvement made to I3 later in the program to handle these failures included an adjustment of the platform visualization itself on the sand table. Rather than try to fix the errors inherent in unreliable automation (a noble, but difficult task given the current state-of-the-art and scope of the program), we decided to place the diagnostic information at the point of individual interaction. Certain data in the telemetry returned by a platform through the dispatcher to I3 could reliably indicate certain types of failures, be it path planning, lower-level ROS errors, or others. Placing this information in the form of a clearly visible icon above the robot model helped the operator quickly recognize that they would likely not get the results they desired if they included the platform in a new tactic.

The problems and findings described above generally fall under the category of high-level tasking failure, and the need to design a system around the types of failures known to occur in real world swarms—unreliable communication, conflicting tasking, and platform hardware or automation failures. Early in I3 development, we focused on designing a high-level tasking interface for swarm automation that we assumed would be present at the exercises, and later found ourselves adding and modifying these lower-level interfaces to allow for more granular tasking. Instead, the correct approach is likely to design these lower-level interfaces first, and then expand them into more general, swarm-level tasking interfaces as the automation matures and we discover more about what works and what does not.

5.4. Speech Recognition and Text-to-Speech

In our initial designs, speech recognition played a large role in I3 interactions. The actual verbal grammar became a driver behind interactions with the swarm. We elected to designate the NATO alphabet (alpha, bravo, charlie, etc.) for both user-generated geometry and dynamic swarm compositions. Along the same theme of consistent recognition grammars, we molded our swarm interactions to use subject-verb-object structure, meaning our verbal utterances and interfaces would begin by indicating the subject of a tactic (platform, swarm, building, area), followed by the action (usually a tactic), target location or building (the objects), and finally any optional additional parameters.

Despite continued effort, poor speech recognition while in the field repeatedly hampered the use of this input modality. A combination of aging recognition engine, mixed quality microphone hardware, and unsuitable audio environments during events led us to deprecate the speech inputs for I3. However, the subject-verb-object structure for tactic composition remained, and proved valuable for I3 operators by providing a predictable mini "checklist" of what each tactic specification required before it could be accepted by the dispatcher.

I3 also used text-to-speech (TTS) to indicate the state of user-invoked high-level tactics, providing an explicit call out when the tactic was initialized by the swarm platforms, as well as the terminal state for completion. We supplemented TTS was other, nonverbal audio cues uniquely associated with different entities and events in the sand table, indicating things like artifact identifications and various scenario-specific interactions between platforms and artifacts. As with other features implemented early in the program, as swarm and tasking size grew, these feedback routines were applied sparingly to avoid overwhelming the operator. In particular, the artifact recognition clip, which we found very useful during sparser scenarios, became an annoyance and a distraction when operating in target rich environments.

Unfortunately, the value of audio cues and TTS proved less important during actual field exercises. The swarm operator required frequent verbal communication with the data collection team member, and often coordinated with safety crew and other command and control center participants. Without a clear design to deconflict these multiples listening demands, and general poor acoustics, the audio feedback was frequently ignored.

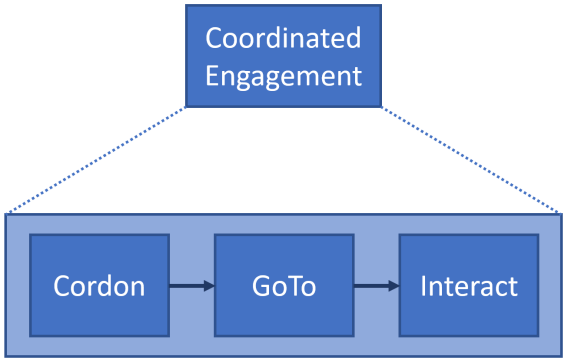


Figure 26. Example tactic decomposition. The `CoordinatedEngagement` I3 tactic represents three chained lower-level tactics, a `Cordon`, where vehicles would surround a supplied point, a `GoTo`, where vehicles would descend from their cordon to a lower altitude, and finally an `Interact`, where the vehicles would attempt to neutralize the supplied hazard.

5.5. Situational Awareness

Treating a large collective of semiautonomous vehicles as a swarm provides a mechanism to increase situational awareness while also requiring increased overall SA to maintain. A swarm, when treated as a single entity, allows the operator to issue a single command, or chain of commands, and use remaining time to survey the environment and monitor the ongoing tactic. Throughout the OFFSET exercises, this increased SA proved invaluable when the swarm broke down, either intentionally or unintentionally, and the operator had to take more direct control of smaller subgroups or individual vehicles and experienced time pressure due to the limited flight time of the aerial vehicles and the quick reaction required to neutralize hazards before they neutralized the platforms. Therefore we designed I3 at all levels of control to provide increased SA to the operator wherever possible. This took the form of both emphasizing high-level actions, through the mission plan, aggregating multiple lower-level tactics into single high-level tactics (Figure 26), and salient notifications highlighting ongoing threats.

Although I3 expressly focused on providing tools to the operator to maintain SA, significant attentional demands in certain circumstances made this goal difficult to consistently accomplish. No better example exists than the issues surrounding single vehicle control, discussed earlier. One of the major factors impacting SA when controlling the swarm at a lower level came from the literal narrowing of the viewport inherent in doing so. Current VR hardware, while significantly improved from earlier iterations, still suffers from lower resolution than human sight can make use of, as well as a much narrower field of view. The closer in to the sand table the operator moves to inspect or direct an individual vehicle, the less of their surroundings they can monitor. We attempted to mitigate this restriction by making use of the auditory channel to highlight to the operator when tactics failed, platforms were neutralized, or new hazards discovered, but the operating conditions of the environment in which the VR equipment existed made this difficult.

Unreliable data presents an additional strain on SA and existed throughout the program. Because practical and logistical limitations required we design, implement, and debug I3 with a simulation as the backend the majority of the time, noisy sensor data proved difficult to account for during the exercises. We discovered a significant operator cost associated with mentally cataloguing the estimated differences between the data reported to I3 by the platforms and what the operator knew to be true based on experiences from earlier in a scenario run. These include GPS drift and general inaccuracy, as well as degraded communications requiring the operator to estimate if a platform stopped due to a crash, hardware failure, or lost connection. Furthermore, interactions between vehicles and hazards relied on Bluetooth Low Energy (BLE) beacons set with fixed interaction radii; however, due to obstacles, reflections, and other environmental features, the distance required

to interact with a hazard varied unpredictably, oftentimes leaving the I3 operator confused as to why they were not able to successfully interact, or why their own platform was neutralized from a seemingly safe distance.

Although there were some mitigations we could take to account for this uncertainty *a priori*, the longer term solution would be to observe and document uncertainty as it occurs and continually update the underlying models to account for this. In I3, the sand table visualization would display the documented threat ranges of various hazards as a ring around the reported position, but this remained fixed throughout the exercise. Under a more adaptive model, the visualization would update as the hazard neutralizes nearby platforms, or as attempted interactions by friendly platforms failed, say by morphing the threat ring to remain consistent with reported outcomes.

The CCAST system resulted in a massive amount of data coming in to I3 at any given point. Typical usage saw hundreds of state updates, hazard sightings, or other updates every second, all of which needed to be distilled into a manageable display that fit into the VR field of view. Therefore we discovered plenty of instances where we, as developers, had to triage in advance for the I3 operator, making decisions that we hoped would improve usability without incurring significant cost in the form of lost situational awareness. Elements that distracted from maintaining good SA included required communication with other actors in the real world, unreliable automation and sensor data requiring the I3 operator to narrow their focus within I3, and more basic roadblocks when developing a novel human swarm interface in virtual reality, such as how to display large blocks of text, video feeds, or other summary data in a manner that did not cover the viewport in flat, 2D displays, thereby nullifying the benefits of immersion VR provides. To summarize our findings succinctly: we determined that swarm interfaces, and VR-based ones specifically, must be designed to encourage engagement at a level appropriate to what the underlying automation and sensor data can realistically support, with an emphasis—but not a requirement—toward higher-level control where possible.

6. Conclusion

This paper presented I3, a virtual-reality swarm control interface developed for the DARPA OFFSET program. I3 provides a single operator the tools to control large numbers of platforms in urban environments at varying levels of specificity—from individual tasking to high-level swarm commands. I3 also presents several tools to aid the operator in maintaining situational awareness throughout a mission, even in the face of mounting vehicle failures, including intelligent aggregation of scenario hazards, visual and auditory notifications, a fluid navigation system, and visualizations for the current mission plan and issued tactics.

Considering the ambitious scope of the OFFSET program, we believe I3 achieved success in providing a single operator meaningful control of a swarm at different levels of granularity. We saw numerous successes in our deployment of a VR-based swarm control interface in the field: the I3 operator was able to make use of the available automation to a high degree, recognize when failures were occurring, and generally achieve at least some of the scenario objectives each mission. Considering this is the first fielding of such an interface within a complex system of swarm platforms, the findings presented above should inspire optimism for future researchers and developers looking to extend this work.

The primary places where future work could improve upon I3 lie in the interaction between the interface and unreliable autonomy and noisy sensor data. Intelligently monitoring the data and developing adaptive models alongside I3 to filter out noisy data and present confidence bounds, for instance when visualizing threat ranges from known hazards, would increase I3's utility and operator SA significantly. Built-in contingencies and aids specifically tailored to situations where the automation breaks down would also assist swarm operators. In cases where a tactic fails, for instance, better reporting of diagnostic data from the point of failure, along with an aid in the interface to reconstruct the tactic—with the details from the previous attempt filled in and ready for modification—would help an operator quickly overcome the failure. Ensuring a more controlled

operating environment would also allow for more utilization of the auditory and haptic channels for feedback, something that we hoped to make better use of throughout OFFSET. Eliminating wasteful routines, better utilization of available sensor data (including uncertainty), and better coordination between interface designers and the automation's developers will all foster growth in human-swarm teaming capabilities in the future.

Acknowledgments

This material is based upon work sponsored by the U.S. Defense Advanced Research Projects Agency (DARPA), Tactical Technology Office (TTO) under contract number N66001-17-C-4067. Special thank you to OFFSET program manager Dr. Timothy Chung at DARPA TTO for making this work possible.

Distribution Statement A: Approved for public release, distribution unlimited.

ORCID

Phillip Walker https://orcid.org/0000-0001-7823-5211
Joshua Hamell https://orcid.org/0000-0002-4657-6203
Jack Ladwig https://orcid.org/0000-0003-0288-5634
Helen Wauck https://orcid.org/0000-0003-2822-4770
Peter Keller https://orcid.org/0000-0003-1413-4758

References

Bandyopadhyay, S., Chung, S.-J., and Hadaegh, F. Y. (2017). Probabilistic and distributed control of a large-scale swarm of autonomous agents. *IEEE Transactions on Robotics*, 33(5):1103–1123.

Bullo, F., Cortés, J., and Martinez, S. (2009). *Distributed control of robotic networks: a mathematical approach to motion coordination algorithms*, volume 27. Princeton University Press.

Chen, M., Zhang, P., Wu, Z., and Chen, X. (2020). A multichannel human-swarm robot interaction system in augmented reality. *Virtual Reality & Intelligent Hardware*, 2(6):518–533.

Chien, S.-Y., Lewis, M., Mehrotra, S., and Sycara, K. (2012). Effects of unreliable automation in scheduling operator attention for multi-robot control. In *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 321–326. IEEE.

Chien, S.-Y., Wang, H., Lewis, M., Mehrotra, S., and Sycara, K. (2011). Effects of alarms on control of robot teams. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 55, pages 434–438. Sage Publications Sage CA: Los Angeles, CA.

Chung, T. H., Clement, M. R., Day, M. A., Jones, K. D., Davis, D., and Jones, M. (2016). Live-fly, large-scale field experimentation for large numbers of fixed-wing uavs. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1255–1262. IEEE.

Clark, S., Usbeck, K., Diller, D., and Schantz, R. E. (2021). Ccast: A framework and practical deployment of heterogeneous unmanned system swarms. *GetMobile: Mobile Computing and Communications*, 24(4): 17–26.

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

Das, A. N., Doelling, K., Lundberg, C., Sevil, H. E., and Lewis, F. (2017). A mixed reality based hybrid swarm control architecture for manned-unmanned teaming (mum-t). In *ASME International Mechanical Engineering Congress and Exposition*, volume 58493, page V014T07A019. American Society of Mechanical Engineers.

Divband Soorati, M., Clark, J., Ghofrani, J., Tarapore, D., and Ramchurn, S. D. (2021). Designing a user-centered interaction interface for human–swarm teaming. *Drones*, 5(4):131.

Gazi, V. and Fidan, B. (2006). Coordination and control of multi-agent dynamic systems: Models and approaches. In *International Workshop on Swarm Robotics*, pages 71–102. Springer.

Goodrich, M. A., Kerman, S., and Jun, S.-Y. (2012). On leadership and influence in human-swarm interaction. In *2012 AAAI Fall Symposium Series*.

Haring, K. S., Finomore, V., Muramoto, D., Tenhundfeld, N. L., Wen, J., and Tidball, B. (2018). Analysis of using virtual reality (vr) for command and control applications of multi-robot systems. In *Proceedings of the 1st International Workshop on Virtual, Augmented, and Mixed Reality for HRI (VAM-HRI)*.

Jang, I., Hu, J., Arvin, F., Carrasco, J., and Lennox, B. (2021). Omnipotent virtual giant for remote human–swarm interaction. In *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*, pages 488–494. IEEE.

Jang, I., Shin, H.-S., and Tsourdos, A. (2018a). Anonymous hedonic game for task allocation in a large-scale multiple agent system. *IEEE Transactions on Robotics*, 34(6):1534–1548.

Jang, I., Shin, H.-S., and Tsourdos, A. (2018b). Local information-based control for probabilistic swarm distribution guidance. *Swarm Intelligence*, 12(4):327–359.

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

Kolling, A., Nunnally, S., and Lewis, M. (2012). Towards human control of robot swarms. In *Proceedings of the seventh annual ACM/IEEE international conference on human-robot interaction*, pages 89–96.

Lewis, M., Wang, J., and Scerri, P. (2006). Teamwork coordination for realistically complex multi robot systems. In *NATO Symposium on Human Factors of Uninhabited Military Vehicles as Force Multipliers*, pages 1–12.

Liu, W. and Gao, Z. (2020). A distributed flocking control strategy for uav groups. *Computer Communications*, 153:95–101.

Mahani, M. F., Jiang, L., and Wang, Y. (2020). A bayesian trust inference model for human-multi-robot teams. *International Journal of Social Robotics*, pages 1–15.

Miller, C., Funk, H., Wu, P., Goldman, R., Meisner, J., and Chapman, M. (2005). The playbook^{TM} approach to adaptive automation. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 49, pages 15–19. SAGE Publications Sage CA: Los Angeles, CA.

Miller, C. A., Draper, M., Hamell, J. D., Calhoun, G., Barry, T., and Ruff, H. (2013). Enabling dynamic delegation interactions with multiple unmanned vehicles; flexibility from top to bottom. In *International Conference on Engineering Psychology and Cognitive Ergonomics*, pages 282–291. Springer.

Miller, C. A. and Parasuraman, R. (2007). Designing for flexible interaction between humans and automation: Delegation interfaces for supervisory control. *Human factors*, 49(1):57–75.

Mondada, L., Karim, M. E., and Mondada, F. (2016). Electroencephalography as implicit communication channel for proximal interaction between humans and robot swarms. *Swarm Intelligence*, 10(4):247–265.

Morris, A. C., Smart, C. K., and Thayer, S. M. (2013). Adaptive multi-robot, multi-operator work systems. In *Multi-Robot Systems: From Swarms to Intelligent Automata: Proceedings from the 2002 NRL Workshop on Multi-Robot Systems*, page 203. Springer Science & Business Media.

Mulgaonkar, Y., Makineni, A., Guerrero-Bonilla, L., and Kumar, V. (2017). Robust aerial robot swarms without collision avoidance. *IEEE Robotics and Automation Letters*, 3(1):596–603.

Nagavalli, S., Chien, S.-Y., Lewis, M., Chakraborty, N., and Sycara, K. (2015). Bounds of neglect benevolence in input timing for human interaction with robotic swarms. In *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 197–204. IEEE.

Nagavalli, S., Luo, L., Chakraborty, N., and Sycara, K. (2014). Neglect benevolence in human control of robotic swarms. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6047–6053. IEEE.

Nagi, J., Giusti, A., Gambardella, L. M., and Di Caro, G. A. (2014). Human-swarm interaction using spatial gestures. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3834–3841. IEEE.

Nam, C., Walker, P., Li, H., Lewis, M., and Sycara, K. (2019). Models of trust in human control of swarms with varied levels of autonomy. *IEEE Transactions on Human-Machine Systems*, 50(3):194–204.

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

Olsen Jr, D. R. and Wood, S. B. (2004). Fan-out: Measuring human control of multiple robots. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 231–238.

Olson, E. (2011). AprilTag: A robust and flexible visual fiducial system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3400–3407. IEEE.

Pourmehr, S., Monajjemi, V. M., Vaughan, R., and Mori, G. (2013). "you two! take off!": Creating, modifying and commanding groups of robots using face engagement and indirect speech in voice

commands. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 137–142. IEEE.

Prabhakar, A., Abraham, I., Taylor, A., Schlafly, M., Popovic, K., Diniz, G., Teich, B., Simidchieva, B., Clark, S., and Murphey, T. (2020). Ergodic specifications for flexible swarm control: From user commands to persistent adaptation. *arXiv preprint arXiv:2006.06081*.

Prorok, A., Hsieh, M. A., and Kumar, V. (2017). The impact of diversity on optimal control policies for heterogeneous robot swarms. *IEEE Transactions on Robotics*, 33(2):346–358.

Rajashekar, S. C., Gururajan, S., Esposito, F., and Ferry, D. (2020). Reconfigurable swarms and multi-user, cooperative uas flights through a virtual reality interface. In *AIAA Scitech 2020 Forum*, page 0737.

Rashid, B. and Rehmani, M. H. (2016). Applications of wireless sensor networks for urban areas: A survey. *Journal of network and computer applications*, 60:192–219.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34.

Şahin, E. (2004). Swarm robotics: From sources of inspiration to domains of application. In *International workshop on swarm robotics*, pages 10–20. Springer.

Sauter, J., Matthews, R., Robinson, J., Moody, J., and Riddle, S. (2009). Swarming unmanned air and ground systems for surveillance and base protection. In *AIAA Infotech@ Aerospace Conference and AIAA Unmanned... Unlimited Conference*, page 1850.

Sauter, J. A., Mathews, R. S., Yinger, A., Robinson, J. S., Moody, J., and Riddle, S. (2008). Distributed pheromone-based swarming control of unmanned air and ground vehicles for rsta. In *Unmanned Systems Technology X*, volume 6962, pages 109–120. SPIE.

Sharkey, A. J. (2006). Robots, insects and swarm intelligence. *Artificial Intelligence Review*, 26(4):255–268.

Walker, P., Amraii, S. A., Chakraborty, N., Lewis, M., and Sycara, K. (2014). Human control of robot swarms with dynamic leaders. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1108–1113. IEEE.

Walker, P., Nunnally, S., Lewis, M., Kolling, A., Chakraborty, N., and Sycara, K. (2012). Neglect benevolence in human control of swarms in the presence of latency. In *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3009–3014. IEEE.

Walker, P. M. (2017). *Improving operator recognition and prediction of emergent swarm behaviors*. PhD thesis, University of Pittsburgh.

Walker, P. M., Miller, C. A., Mueller, J. B., Sycara, K., and Lewis, M. (2019). A playbook-based interface for human control of swarms. In *Human Performance in Automated and Autonomous Systems*, pages 61–88. CRC Press.

Weingart, T., Graham, P., Christman, D., and Weingart, A. (2018). Using virtual reality to control swarms of autonomous agents. In *Proceedings of the 20th International Conference on Multimodal Interaction: Adjunct*, pages 1–4.

Winfield, A. F. and Nembrini, J. (2006). Safety in numbers: Fault tolerance in robot swarms. *International Journal on Modelling Identification and Control*, 1(ARTICLE):30–37.

Winfield, A. F., Sa, J., Fernández-Gago, M.-C., Dixon, C., and Fisher, M. (2005). On formal specification of emergent behaviours in swarm robotic systems. *International journal of advanced robotic systems*, 2(4):39.

How to cite this article: Walker, P., Hamell, J., Miller, C., Ladwig, J., Wauck, H., & Keller, P. (2023). Immersive interaction interface (I3): A virtual reality swarm control interface. *Field Robotics, 3*, 605–636.