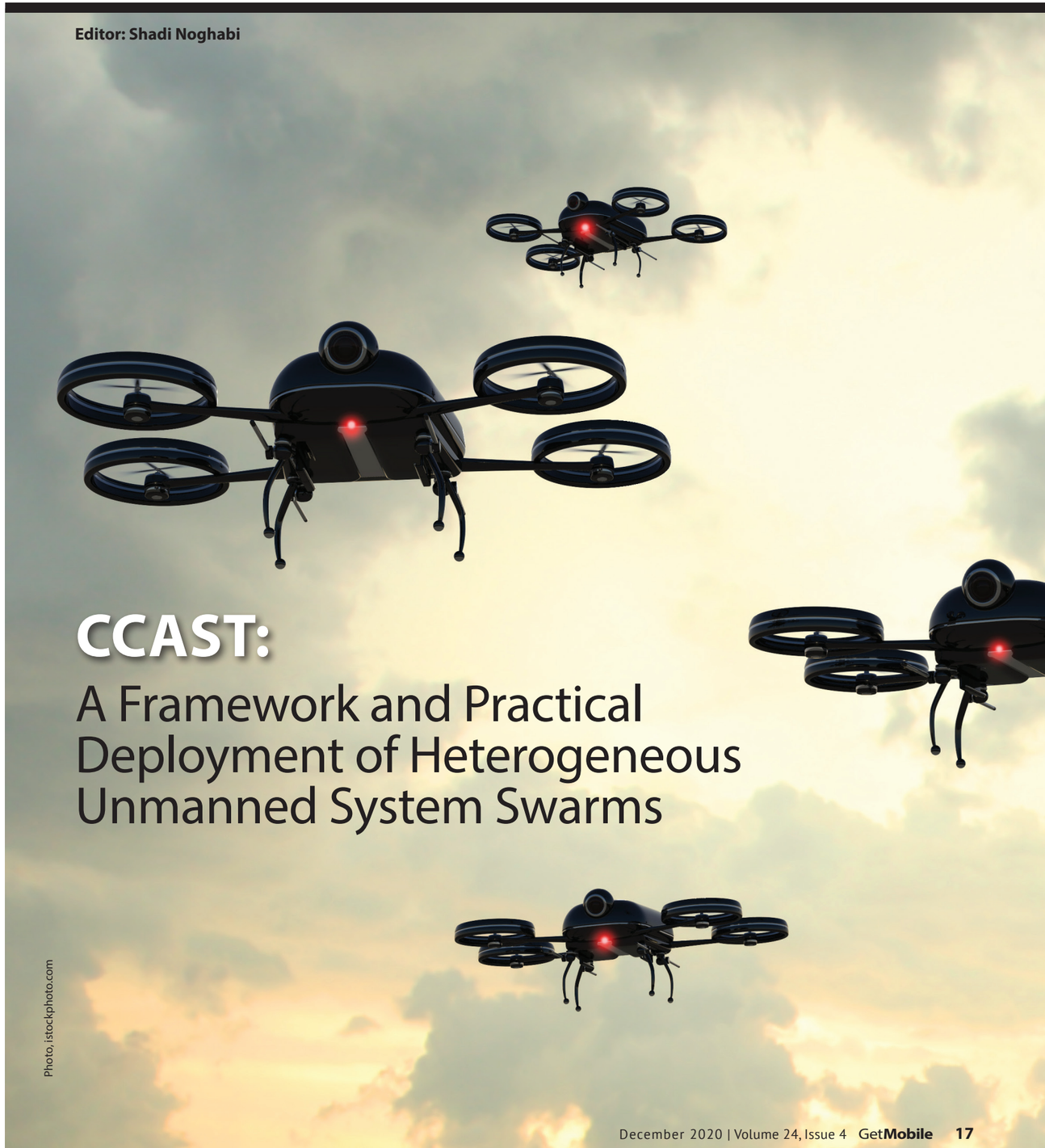
Shane Clark, Kyle Usbeck*, David Diller and Richard E. Schantz *Raytheon BBN Technologies, Cambridge, Massachusetts*
** Participated in this research while employed at Raytheon BBN Technologies*

Editor: Shadi Noghabi



CCAST:

A Framework and Practical Deployment of Heterogeneous Unmanned System Swarms

Photo, istockphoto.com

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA).

Individual semi-autonomous unmanned systems have already proven to be useful in a variety of use cases including movie production, agriculture, civil engineering, military operations and insurance [1]. Likewise, small groups of unmanned systems have shown promise, particularly when coordinating to achieve a common goal [3]. Less studied are the potential benefits of large swarms comprising hundreds of capable unmanned systems, likely due to the difficulties associated with creating, or even simulating, such a system. Swarming unmanned systems operate in an environment with a large number of simultaneously moving vehicles, with extensive coordination in real time using mobile communication over a rapidly shifting and dynamic environment. They are often operating in confined areas, moving at speeds of several meters per second, and combining outdoor and indoor coordination.

In this paper, we describe our practical approach to designing and implementing a system for enabling the organization, planning, and deployment of a large-scale heterogeneous swarm of unmanned air and ground robots within a common mission-oriented framework. Within that framework, we develop and integrate methods, common services and swarm tactics for exploration, evaluation and refinement. We have built an extensible, collaborative swarm ecosystem called Command and Control of Aggregate Swarm Tactics (*CCAST*), including swarm simulators of varying fidelity, with swarm mission code interoperable under simulation and real mission engagement. We describe how CCAST was used to plan, deploy, and evaluate a realistic, complex swarm mission in recent field experiments and highlight the outcome of those experiments. The CCAST system undergoes regular and repeated evaluation exercises to measure progress. These experiments are a means of evaluating the costs and benefits of drone swarm capabilities and demonstrating the effectiveness of CCAST methods in supporting them.

The CCAST framework is being developed as part of the DARPA Offensive Swarm-Enabled Tactics (OFFSET) program, which envisions using swarms comprising upwards of 250 unmanned aircraft systems (UASs) and/or unmanned ground vehicles (UGVs) to accomplish diverse missions in complex urban environments [2]. Our recent exercise successfully engaged up to 60 heterogeneous, air and ground platforms in mission execution.

By leveraging and combining emerging technologies in swarm autonomy and human-swarm teaming, the program and project seek to enable rapid development, evaluation, and deployment of leap-ahead capabilities.

RELATED WORK

Drone swarms are a topic of much current research and experimentation, but it is a relatively new area and lacks consensus around established technology or best practices. There are many dimensions of related work encompassing CCAST. We briefly summarize a few prominent ones for perspective on our current platform, capabilities, and experimental results.

At the platform level, we are creating a *distributed system* for organizing and controlling a large scale, coordinated collection of heterogeneous drones, including air and ground platforms intended to accomplish complex, multi-dimensional and multi-phase missions. Aspirations for large scale are multiple hundreds, in deliberate incremental steps (recently about 75 coordinated vehicles). That degree of scale by itself presents numerous choices and challenges and mandates new solutions over single drone capabilities in areas such as plan decomposition, coordination and communication in close and sparse mobile operations, launch and flight logistics, and dynamic failure management. Deconfliction, coordination and dynamic failure management over fast moving independent platforms obsoletes the solutions offered for single and small drone configurations. Prior related work had fewer platforms [4], a single coordinated, non-reactive objective [6], or staged laboratory experiments with less capable platforms [8]. Beyond the scale and scope of the end mission, our integration and coordination software represents a distributed middleware dynamically incorporating a multitude of services for organizing, selecting, communicating, monitoring, and coordinating individual platforms into a cohesive whole. We share similar objectives to other emerging swarm software frameworks, for example [9] and [10]. These platforms use different approaches to establish a common, reusable middleware framework, focused on different visions for applications and uses. However, ours is the first to specifically focus on contributions to harnessing the power of very large-scale swarms.

At higher levels, our platform incorporates functions and services enabling complex, large scale, coordinated missions, including planning, group communication, obstacle avoidance, and contingency handling. There is much ongoing work in these subareas complementary to services in our platform, for example [11]. Much of this work is theoretical and focused on near-optimal outcomes under selected constraints. Our services work is focused on demonstrating the utility of simple forms that are operationally capable of supporting our large-scale experimentation. It includes companion work on interfaces allowing a single mission commander to provide oversight of a semi-autonomous mission at very high levels of abstraction [18].

CCAST INTEGRATED CONCEPT AND DESIGN

System Design

Designing a control framework to manage a large number of autonomous, mobile agents in complex urban environments requires careful consideration and balancing of the human, computational, and networking constraints inherent in live hardware testing.

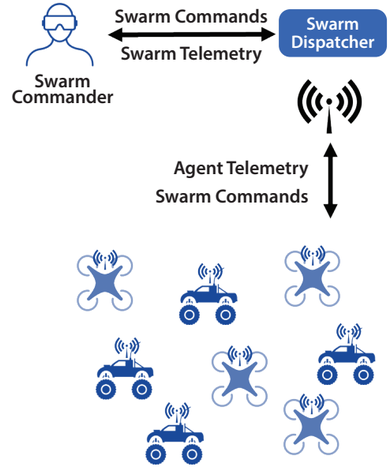


FIGURE 1. In the CCAST network architecture, a human operator controls the swarm from a single command point that receives all telemetry from the agents. In large and complex environments, some agent telemetry may be delayed by seconds and other agents may be completely disconnected from the network for minutes at a time while moving through areas of poor coverage.

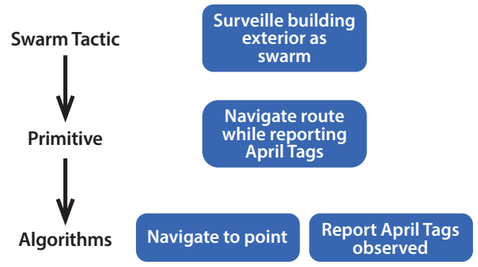


FIGURE 2. The CCAST swarm tactics hierarchy: Lower layers represent the basic building blocks designed for reuse and combination into collaborative, higher-layer "tactic" behaviors.

1) Human-facing design

All swarm tasking in CCAST currently originates from a single human operator to minimize the potential for dangerous or confusing miscommunications. The human operator, known as the Swarm Commander, specifies commands using human-centric names for behaviors, e.g., "surveil building 17 with sub-swarm Alpha." A designated Swarm Dispatcher node receives commands and transforms them into a series of agent-friendly subcommands, often by decomposing the problem into specific agent actions and assigning agents to those actions, as shown in Figure 1. In this example, the Swarm Dispatcher would (for one implementation of the survey tactic): calculate the set of poses necessary for agents to completely cover the convex hull of the object labeled Building 17, split those points evenly among the agents referred to as Alpha, and send each of those agents a set of poses to visit. The agents themselves are responsible for automatically choosing the order in which to visit the poses and for running the decentralized path planner used to deconflict airspace.

Allowing a single human to control up to hundreds of agents simultaneously is a major CCAST functional requirement, along with preventing the human from being overloaded by the sheer volume of information produced by agents. Our approach to this problem is to leverage a virtual reality interface (the Immersive Interactive Interface, or I3) that embeds the Swarm Commander in a 3D sandtable environment, where the swarm telemetry is rendered into virtual agent representations with which the Swarm Commander can interact using VR wand controllers (see Figure 3). The Commander can freely change perspective in the environment to track groups of agents as they move through and interact with the environment. While the VR interface helps avoid operator overload, it is also necessary to provide the Commander with an efficient means of coordinating the swarm at mission levels. CCAST provides command automation support that relieves the user from needing to perform tedious operations, such as individually selecting agents or monitoring battery levels. Instead, the Swarm Dispatcher node automatically selects agents based on location and capabilities, and automatically coordinates low battery replacements, including offloading tasks in progress. These functions allow a Commander to delegate common low-level tasks while focusing on higher-level goals like choosing buildings to explore or interacting with dynamic experiment infrastructure.

CCAST supports a priori planning via a special mission planning component called Swarm Tactic Operations Mission Planner (STOMP). STOMP seeks to ease the Swarm Commander load at mission execution time by providing an interface to pre-plan sets of commands, group them, and specify signals to trigger those groups of commands. A common mission plan element, for example, is a signal that triggers the simultaneous surveillance of many buildings at mission start. The Swarm Commander can send this signal and then be undistracted by further tasking as the surveillance results become available.

While many of the swarm-level design decisions are designed to support and interface with a human commander, CCAST also presents a swarm tactics architecture intended primarily for developers authoring new swarm behaviors. CCAST splits the swarm tactics architecture into three abstraction layers as a means of controlling complexity and enabling reuse, as depicted by Figure 2. At the lowest layer of the hierarchy are algorithms, which are basic agent behaviors required to function as a swarm element, such as processing captured images to look for April Tags. Primitives are the second layer. A primitive is a potentially multi-agent behavior that is still too rudimentary to be considered a collaborative swarm task. An example primitive is the implementation of moving to a given set of points in space. The highest abstraction layer is the swarm tactic, an explicitly collaborative behavior that achieves some tactical purpose

for the Swarm Commander. Tactics and compositions of those tactics are generally the only elements of the hierarchy that an operator directly uses to task agents. Example swarm tactics CCAST implements include building surveillance and patrols in a flock formation.

2) Platform-facing design

To operate safely and provide a commander with sufficient context to complete a mission, a swarm needs to perceive obstacle locations and major environment features. While single, or limited numbers of agents, are typically designed to operate autonomously and independently, our swarm platforms must be physically small and inexpensive to make large-scale operations in congested, dynamic environments feasible. To that end, we choose to rely on the effective combination of low cost sensors. As an example, all agents in the swarm run a decentralized path planner based on the RT-RRT* algorithm [14] that considers the reported location of all agents rather than attempting to detect others with an expensive local sensor, such as spinning lidar. These reported locations are only timely and accurate for those agents currently connected to the swarm network. This design choice results in truly swarm-oriented perception – disconnected platforms cannot operate safely in the presence of others because they cannot "see" them. CCAST currently implements a configurable behavior in this case, either stopping movement to minimize collision probability or continuing to operate unsafely to maximize mission completion probability. Each agent, at a minimum, also comes equipped with a low cost monocular camera used to read April Tags [16] in the environment and shares only the associated semantics as opposed to raw imagery.

We designed the CCAST communications paradigm under the assumption that limited network throughput and reliability represent key constraints for large robot deployments in complex environments. In light of these limitations, each swarm agent periodically shares a small telemetry package via the network (see Figure 1). This package includes the agent's UID, position, heading, battery level, and current task – providing enough information for the human operator to reason about swarm state and for other agents to use reported position for path deconfliction. Telemetry messages are limited to only critical information to conserve bandwidth and are streamed via UDP to minimize the impact of transient disconnections as agents move around large buildings and other environment obstacles. Other messages that agents send as part of specific behaviors are disseminated using lightweight publish/subscribe methods that minimize unneeded network propagation.

CCAST swarm agents sometimes coordinate implicitly and sometimes explicitly depending on the behavior. Implicit coordination is often the result of the centralized Swarm Dispatcher node creating a plan that is conflict-free, e.g., when splitting up and assigning surveillance points to a set of platforms monitoring an area. Explicit coordination is necessary for dynamic or ongoing tasks where next actions are determined by fresh information about other swarm agents. CCAST also includes a custom, lightweight publish-subscribe framework to support the straightforward addition of arbitrary coordination messages for specialized behaviors. A resilient "follow the leader" behavior is one example where explicit coordination is required. CCAST uses publish-subscribe for leader election and leader position updates on a continuous basis.

Scalable swarm operations require that both hardware and software solutions be minimal to control costs and resource consumption as the number of agents increases. We have made a number of tradeoffs in the CCAST design that seek to balance agent independence with linear, or ideally sub-linear, growth in the number of agents. Agents that do not need to navigate indoors and run heavy processing loads such as Simultaneous Localization and Mapping (SLAM) require nothing more than a Raspberry Pi 3b ($35) [19] or better as a co-processor running our agent autonomy stack and a camera such as the Pi Cam ($25) [20]. The major tradeoff in this case is the necessity of maintaining a live network connection for inter-agent path deconfliction. A platform capable of independent safe movement would require more expensive hardware, such as a 3D lidar and a more powerful co-processor to handle the incoming data in near real time.

Swarm deployments including tens to hundreds of flying platforms are potentially dangerous and cannot be rendered safe

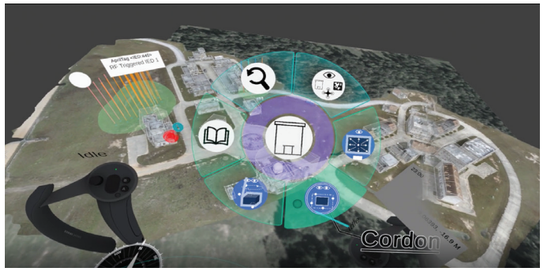


FIGURE 3. The I3 virtual reality interface presenting a radial context menu for building selection.

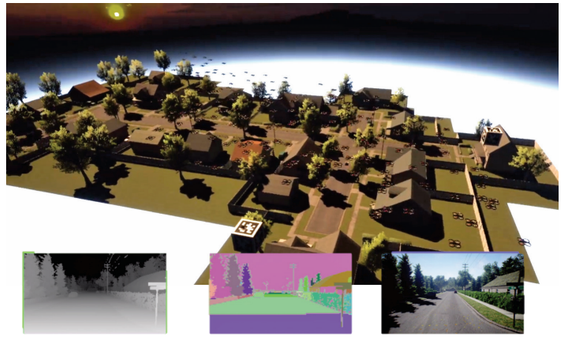


FIGURE 4. Screenshot of 200 autonomous UAS in the CCAST simulator; subwindows respectively show the depth map output, image segmentation, and EO camera output from a given platform.

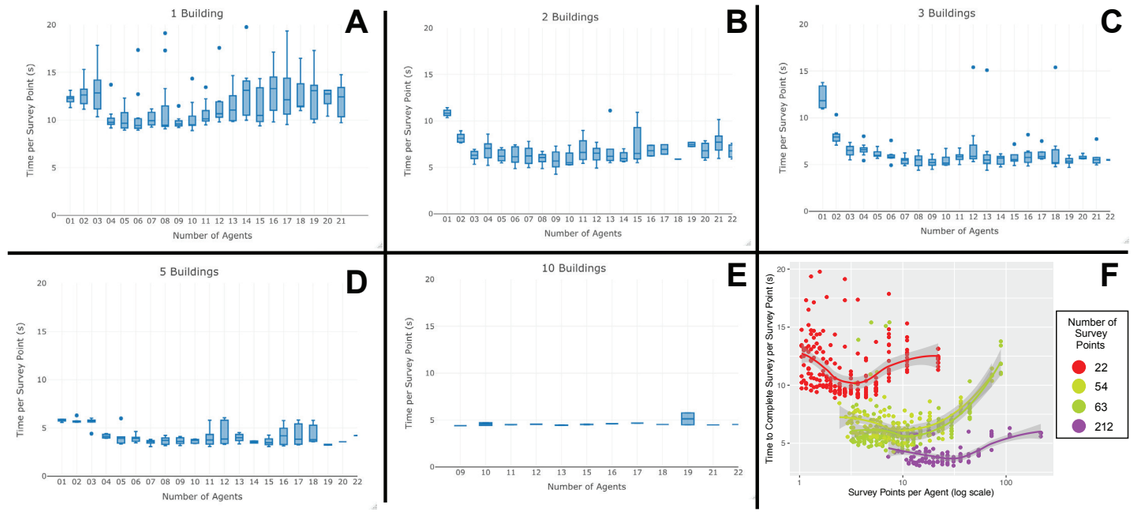


FIGURE 5. Graphs showing the time required to complete a building surveillance task (normalized by the size of the survey) vs. number of agents used. (A-E) show per-building results and (F) shows the normalized aggregate. Note that building surveillance tasks are decomposed into "survey points," which specify the set of platform poses required to completely survey the set of buildings specified.

easily by human-directed overrides. Even with enough humans to manually pilot all platforms, no commercial platform uses an RC controller capable of operating without interference in the presence of so many transmitter/receiver pairs. Given the infeasibility of manual control, we have implemented a multi-layered safety protocol to prevent dangerous situations. The CCAST agent logic will reject any commands that would cause it to leave a pre-defined "region" boundary, fly below a minimum altitude, takeoff without passing safety checks, or exhaust its battery without landing at its original launch location. In case of a total CCAST software crash, we also configure all agent autopilot modules with slightly less conservative geofencing and battery failsafe parameters to maximize the chance of safe return under a variety of failure conditions.

INTEGRATED SIMULATION AND REAL TIME ENVIRONMENT

The cost and complexity of evaluating swarms necessitates the use of simulation for testing and evaluation, prior to live assessment. Simulation is critical for debugging, exploration, and evaluation at multiple levels, from swarm algorithms, to primitives, to tactics, to large-scale mission plans. While there exist high fidelity simulation environments designed for single or small numbers of agents [15, 17], there were no high-fidelity simulators for swarms, with existing solutions severely limited in their scalability. We developed the CCAST Simulator to fill this need.

We extended AirSim, a game-based, open-source simulator for unmanned vehicles, built on top of the Unreal Engine 4 game engine. AirSim has a state-of-the-art physics engine that can simulate individual quad-copter rotor movements, time of day and weather effects, and produce photo-realistic environmental models [17]. AirSim was designed to provide an environment in which robotics systems developers can generate large amounts of data for training machine-learning algorithms involved in autonomous vehicle operation. The project focus was on single vehicle operations. We extended AirSim for the simulation of swarms (tens to hundreds) of UASs and UGVs, and to additional sensors and environmental domains, matching those we are testing in live field exercises. Figure 3 depicts a screenshot of a simulation game-level from the CCAST Simulator.

The CCAST Simulator is architected to integrate with autonomous vehicle flight control systems, including Pixhawk [23], using the MAVLink protocol [24]. Thus, the CCAST Simulator can perform either hardware-in-the-loop simulation or software-in-the-loop simulation. It is also possible to distribute the simulation load across multiple hosts by executing platform logic and control components separately from the physics simulation and visualization. The human-machine interfaces have also been abstracted to allow seamless transitions between simulation and real-world operations. This allows users to build expertise with interfaces prior to live exercises, and evaluate interfaces through simulation as well.

Because of the high computational requirements of high-fidelity simulation with swarms of agents, the CCAST simulator supports three levels of simulation fidelity: *heavyweight, lightweight*, and *featherweight*. Heavyweight simulation provides the highest fidelity, with software-in-the-loop autopilots and full per-rotor physics simulation. In lightweight mode, the software-in-the-loop autopilots and full physics simulation are replaced with a lightweight simulation that receives MAVLink commands just like the autopilots, but immediately performs those commands in an idealized way. Featherweight mode has the least fidelity and

computational load, running without AirSim and its simulated sensor streams. To assist with executing high-fidelity simulation over large swarms, the load can be distributed across multiple computers, running multiple instances of AirSim, with simulated video stream generation occurring on separate compute resources.

Simulation has been extremely useful for:

- Debugging, testing and evaluating algorithms and tactical behaviors for individual agents and swarms.
- Comparing different variations of algorithms and tactics.
- Assessing system performance under infrequent, black-swan type events.
- Evaluating the efficiency and effectiveness of different mission plans.
- Assessing mission performance using different configurations of agents and agent capabilities.

A. Use of Simulation to Advance the Design of Swarm Services

CCAST provides services and tactics to simplify swarm system mission preparation, deployment, and monitoring. Swarm tactics, which describe scalable collaborative behaviors of CCAST agents, often require configuration parameters (such as how many platforms to marshal for a particular coordinated service, or how long to loiter looking for something of interest) to be initially set at instantiation-time. These parameters can have a huge impact on the efficiency and effectiveness of the tactic deployment. To derive and embed appropriate parameters into the mission-planning tool, we use a Monte Carlo tactic tuning method to modify various conditions and rapidly execute them in the CCAST Simulation Environment.

In this section, we examine the use of the tactic tuning method toward automating parameterization of the number of agents to employ in mission plans containing the Building Surveillance Tactic, illustrating the synergetic use of simulation capabilities for designing and improving CCAST's automated services.

B. Building Surveillance Example

The ideal number of agents to use in surveying collections of buildings depends on many different attributes. To experimentally determine numbers of agents, we configured the CCAST simulator to best match the Camp Shelby CACTF environment, shown in Figure 6.

The Tactic Tuning Method automatically executed 912 simulations with various parameterizations allowing each to run to mission completion. The runs measured the time to complete each mission while varying the number of agents to deploy (from 1-22) and the number of buildings to survey (1, 2, 3, 5, and 10), while keeping all other variables constant.

The building surveillance tactic works by decomposing 3D building polygons from the model into numerous *survey points* (i.e., geospatial poses). By loitering at this set of survey points, platforms can effectively surveil the entirety of building convex hulls. The more and the larger the buildings, the more total survey points required to achieve complete coverage.

Figure 5 shows results from a Tactic Tuning Service experiment visualized as the mean time to complete surveys versus the number of agents used. Panels A thru E show results for each of 1, 2, 3, 5 and 10 buildings surveyed together. The Y-axis is normalized by the number of survey points in order to visually emphasize natural groupings based on the number of buildings. Figure 5F shows the composite view of all the runs together, again normalized by the number of survey points. Each color represents the runs for a particular building configuration: red for 1 building, yellow for 2 buildings, green for 3 buildings, and purple for 5 buildings. The runs with 10 buildings were dropped from this figure because the low mission success rate did not provide enough data.

These charts show two important results. First, the number of buildings in a surveillance run impacts the time to complete the tactic independent of the total number of points, as shown in Figure 5F. This is due to the grouping of points in space – those around the same building are generally closer together than those around distinct buildings, which minimizes transit time among them. Second, these graphs show that the mean time to visit each survey point tended to decrease with increasing numbers of agents, up to an inflection point. Assigning more platforms to the building surveillance meant the total number of survey points needed to be covered for a given configuration of buildings were completed in less elapsed time, until such time as there were "too many" in a confined space. At that point, collision avoidance maneuvers cost more time than the increased parallelism gained. Therefore, there exists an upper bound for a range of potential platforms to use, beyond which we generally take longer. Further, the range is expected to be a function of the number of survey points, which can be estimated beforehand. Figure 7 illustrates the detrimental effects of exceeding the upper bound by showing the percentage of successful missions as the number of agents used increases. Increased contention for airspace quickly leads to skipped survey points because agents cannot find safe paths to reach them.

We used these results to derive appropriate parameters to embed into automated range selection for the surveillance tactic.

FIGURE 6. Map of the Camp Shelby CACTF annotated with deployment locations and buildings chosen for surveillance.

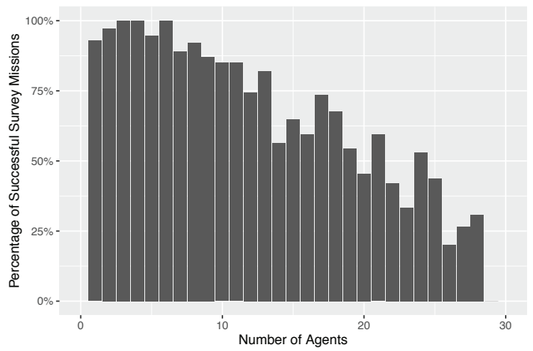


FIGURE 7. The percentage of missions successfully tasking platforms to each survey point vs. the number of platforms deployed.

FIGURE 8. Camp Shelby Experimental Range.

FIGURE 9. Example of April Tag positionings at FX-3.

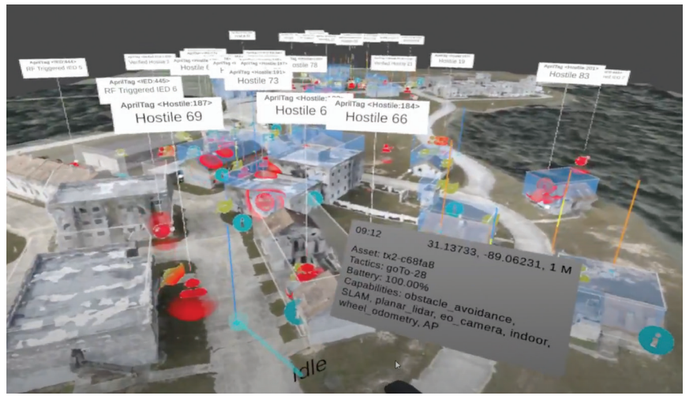


FIGURE 10. The I3 virtual reality interface uses the georectified 3D model of the area of interest, enabling the Swarm Commander's increased situation awareness while controlling the swarm.

EXERCISE DESCRIPTION AND EVALUATION

Regularly testing swarm systems in the field is crucial for meaningful evaluation. Mobility in a complex environment inevitably produces effects that are notoriously difficult to simulate at scale, especially with respect to radio propagation for localization and communication. These effects in turn are likely to expose new or different failure cases in the layers that build atop these fundamental building blocks.

To ensure that our development efforts remain realistic and measure progress, the OFFSET program conducts field exercises approximately every six months. These exercises are of increasing scope and complexity, including growth in swarm size (50 to 250 vehicles), area of operations (two city blocks to eight city blocks square), and mission duration (15 minutes to 4-6 hours).

For two weeks in December 2019, our CCAST team worked with a government evaluation team and five teams of technology developers at the Camp Shelby Combined Arms Collective Training Facility (CACTF) near Hattiesburg, MS. This was the third scheduled field exercise (FX-3) for the OFFSET program.

The Camp Shelby CACTF (Figure 8) is a 2.25 km^2 realistic urban warfare training facility consisting of 26 buildings of various heights and dimensions, paved and unpaved roads, alleys, fields, and foliage.

The goals of FX-3 were to deploy and test CCAST swarm command and control systems within a government-team-developed scenario targeted at rewarding large scale coordination in a realistic urban environment. The overall goal of the scenario was to safely find all of the "high value" objectives within the area of interest.

Some scenario elements were virtualized to expedite experiments and improve evaluation. For instance, scenario information was scattered around the CACTF with April Tags (9) to focus our agents' information discovery. Similarly, our agents interacted with Bluetooth Low Energy (BLE) beacons in the environment to simulate platform loss and information discovery (e.g., agents finding high value pieces of information). Heterogeneous agent collaboration was encouraged through a number of scenario intricacies.

A. Operational Phases

To illustrate how CCAST services are used to organize, compose and execute complex swarm missions, we will consider ordered phases of operation:

1) Initial Survey of the Area of Operations

CCAST first surveys the mission area with high-resolution, high-accuracy mapping platforms. The output of this stage is a geo-rectified 3D model from which we derive:

- A high-resolution georectified ortho-mosaic for 2D situation awareness displays.
- Digital terrain and surface models.
- A categorized point cloud providing high-level navigation guidance to plot near-optimal courses avoiding buildings, cliffs, etc., while preferring advantageous terrain.

The 3D model also provides realistic scenes for our simulator and background for the virtual reality swarm control interface, I3 (10).

2) Mission Planning

CCAST provides the interactive Swarm Tactics and Operations Mission Planner (STOMP) tool for building mission plans (see Figure 11). STOMP integrates with the multi-resolution Swarm Simulation environment to rapidly test, evaluate, and refine those plans, providing near-real-time insights to the mission developer through STOMP. Mission plans can be persisted and shared between instances of STOMP, enabling plans to be created long before mission start, but modified right until they are *instantiated and launched.*

3) Instantiating the Mission Plan

Once platforms have been staged for launch and powered on, the operator loads the pre-configured mission plan. The plan is instantiated using a constraint optimization approach by binding available platforms on the network to roles or groups in the plan. Once fully instantiated, the Swarm Commander enters the virtual reality I3 interface.

4) Controlling the Swarm During Operation

Even the largest CCAST swarms are intended to be overseen by a single Swarm Commander. I3 allows commanders to visualize and control execution of mission plans. The Swarm Commander monitors progress from within I3 and triggers additional actions responding to unfolding scenario information.

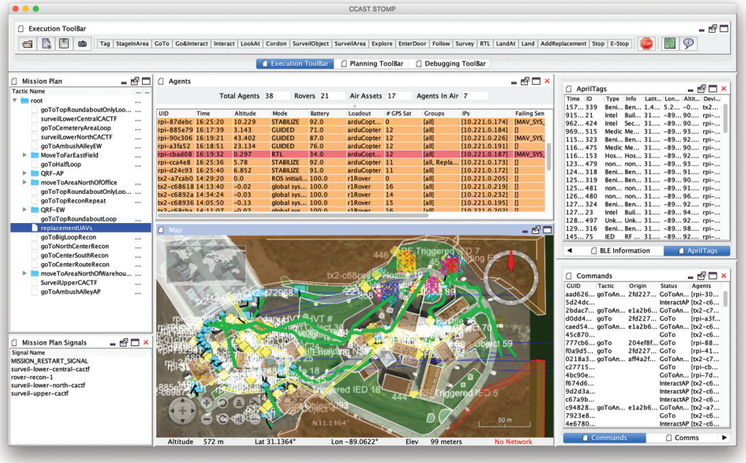


FIGURE 11. The Swarm Tactics Operations and Mission Planning (STOMP) tool.

5) Automatic Mission Execution Monitoring

During the entire mission, the CCAST framework automatically monitors each platform and overall progress against the plan. Each platform's autonomous agent logic constantly monitors platform health and the status of other nearby agents along with their progress towards shared goals. Agents can also dynamically request help from other platforms. For instance, a platform sensing low battery will automatically request a replacement to pick-up where it left-off; or a platform that encounters a condition or threat will request backup from a better positioned agent to manage the detected threat.

6) Safety and Remote Monitoring

The positions and status of CCAST platforms can be monitored both on-site (e.g., by safety spotters) and remotely in near-real-time, using a variety of displays, ranging from heads-up augmented reality interfaces, to a common Android phone running the Android Team Awareness Kit (ATAK) for situation awareness [22].

B. Exercise Observations

Agents autonomously respond to environmental stimuli, real (e.g., encountering an unexpected obstacle) and simulated, and report these events back to the swarm dispatcher which aggregates, logs, and forwards portions to an exercise evaluator's framework. Given many simultaneous streams of agent telemetry, and the lack of definitive context for many common events (e.g., why an agent followed a given path plan), thorough quantitative assessments for swarm exercises are still works in progress. At current maturity, we instead rely on a combination of readily understood aggregate metrics, such as swarm scale, and empirical observations of high-level behaviors. Defining and accurately capturing new swarm metrics is ongoing research to enable baselines as future points of comparison.

1) Coordination

CCAST implements primitives and algorithms for various forms of multi-agent coordination including consensus and deconfliction. The coordination algorithms are of two approaches: 1) centralized coordination where an explicit pre-agreed-upon agent makes a decision for the group, or 2) decentralized coordination where a dynamically elected leader makes those decisions. Decentralized coordination offers various election strategies and re-election criteria. Decentralization has higher computational and communication overhead, but effectively eliminates single points of failure. For the FX-3 evaluation exercise, most CCAST tactics used a centralized coordination model because the underlying network (LTE) was centralized. In that configuration, distributed coordination offered little advantage because loss of a backhaul link stopped all communications. During the experiment, there were no aerial platform-on-platform collisions. Throughout the exercises, whenever a platform issued an automated call for support or replacement (e.g., when low battery prevented completing an assigned task), it was answered with an additional or replacement platform until we had exhausted all platforms held in reserve.

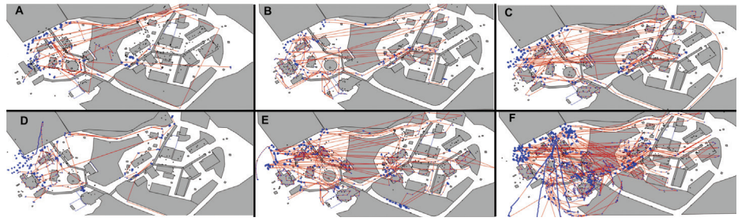


FIGURE 12. Increasing scale of vehicles deployed and area covered.

2) Navigation

Environmental obstacle avoidance worked well throughout the exercise, with some exceptions. Most environmental collisions occurred with powerlines and road signs – i.e., small items that were not captured well in 3D models and needed manual addition as obstacles. Dynamic obstacle avoidance functioned fairly well, and was likely overly conservative, as we experienced no collisions, but did experience deadlock conditions where platforms could not find safe paths around other platforms, requiring manual intervention.

Ground rovers have more constraints on their motion, so safe path planning is more complex. Our rovers struggled to navigate over some terrain – particularly muddy areas, necessitating preference for paved roads in path planning. Since our navigation pipeline offers characterization of point clouds, it was straightforward to add this preference.

CCAST's GPS-denied SLAM stack enabled rover exploration of indoor spaces. On several occasions, rovers successfully entered buildings and explored portions of the floorplan. However, we often lost network connectivity when entering the concrete buildings. In consequence, determining how well the rovers functioned indoors was severely hampered. In the few cases where a rover made it back outside after exploring a building, our disruption-tolerant networking successfully queued and sent information collected during their disconnected interior exploration.

3) Scale

As the on-site exercise period progressed, we were able to deploy larger-sized swarms, cover more area, and penetrate more complex environments (e.g., exploring GPS-denied areas). Figure 11 shows maps of the area of interest where each panel represents a different experimental run with platform location reports (represented by blue dots connected by red lines). With each new experimental run, panels A through F show a general increase in area covered and larger numbers of platforms deployed. In our largest run, the CCAST team deployed more than 60 platforms in the swarm and found more than 180 unique April Tags.

LESSONS LEARNED AND NEXT STEPS

The CCAST team has learned many lessons in the course of development and also identified a number of open questions. One lesson learned is that there is a minimum set of capabilities necessary for swarm agents to interoperate safely, which includes detection of other agents with a sufficient time horizon to plan and execute safe trajectories in response. This capability can be satisfied by information exchange (as in CCAST), direct detection, or centralized control, with tradeoffs among cost, size, and network requirements associated with each approach. Another key lesson is that operator overload must be addressed as a first order concern in swarm systems, before optimizing for operational efficiency. Platforms affordable enough to scale to large swarms do not have the bulky, high-fidelity sensors or power-hungry computational resources necessary for long-term unsupervised autonomy, and thus require the human commander to have persistent and reliable situation awareness at a glance. Key open questions remain about how to design and build swarm systems suitable for a range of possible operating environments. Large open spaces where agents can operate at high speeds and maintain conservative safety distances call for very different design choices than the congested pseudo-urban environments in which we have tested CCAST. The operator's environment is also a key parameter for system design. CCAST's VR interface is optimized for immersion and multi-modal inputs, but other contexts require a "heads up" display or for the operator to have free hands. The choices made for CCAST's I3 component cannot simply be ported to fundamentally different form factors, and adapting to other constraint sets will require further work on identifying the set of generalizable requirements for such a user interface.

CCAST is currently under active development, with ongoing goals of increasing scale to 250+ simultaneously deployed platforms, continuing to simplify and refine integration points, and increasing the level of closed-loop autonomy – for both individual agents and the swarm as a whole. As these three technical thrusts co-evolve, we expect to gain further insights into the management of cooperative, heterogeneous semi-autonomous systems at previously untested scales.

CONCLUSION

We have introduced and described at a high level the elements of the CCAST drone swarm framework for organizing large collections of autonomous platforms into a cohesive mission-focused capability. It is a comprehensive set of methods, services, capabilities and means for evolution and extension, going from organizing, testing and executing drone swarm tactics to complete missions cooperatively executed

by large collections of aerial and ground platforms. To our knowledge, it is the most comprehensive framework available to explore the benefits and costs of automated and semi-automated large-scale drone technology. Live field experiments have shown the viability of the CCAST approach toward organizing and supporting drone swarm activity, and demonstrated upward of 60 independent platforms operating together under the control of a single mission commander.

Experimentation included integration with five external collaborators, demonstrating the utility of the CCAST architecture. We have taken a number of steps to ensure that externally developed capabilities can interoperate and integrate effectively into our swarm ecosystem. 1) We have developed an architecture that is open and extensible, including interfaces with commonly used standardized tools (e.g., Robot Operating System (ROS), PX4, ArduPilot). 2) We have designed simulation and hardware interfaces to maximize transparent code reuse between the two. 3) We have developed CCAST training materials to assist with development and interoperability issues. We encourage outside organizations to consider integration and evaluation experiments with CCAST in the future.

A video summarizing the FX3 CCAST drone swarm exercise is available online at https://youtu.be/00jUHD3LBsQ ■

Acknowledgment

The authors would like to acknowledge DARPA sponsorship of this activity, in particular the program manager, Dr. Timothy Chung, the government evaluation team, Naval Information Warfare Systems Command (NIWC) Pacific, and all of the technologist participants. The views, opinions and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

Shane Clark is a Senior Scientist at Raytheon BBN Technologies, where he leads the Unmanned Innovations Lab. His recent work focuses on swarming and collaborative autonomous systems. He received his PhD at the University of Massachusetts Amherst in 2013. shane.clark@raytheon.com

Kyle Usbeck is a Lead Scientist and Engineer at Systems & Technology Research (STR), where he specializes in collaborative autonomy. Previously, he worked at Raytheon BBN Technologies, where he founded the Unmanned Innovation Lab and led research and development projects involving unmanned autonomous systems. He holds a MS and BS in Computer Science from Drexel University, and is a senior member of the ACM and IEEE. kyle.usbeck@stresearch.com

David Diller is a Lead Scientist and Group Lead at Raytheon BBN Technologies, where he is involved in projects at the intersection between people and complex systems, including simulation and game-based training and tutoring, autonomous systems, and situational awareness systems. He has a Masters in Computer Science and a PhD in Cognitive Psychology. david.diller@raytheon.com

Richard Schantz is a Principal Scientist at Raytheon BBN Technologies, where he serves as a technical advisor to projects pursuing advanced research in distributed systems technology. He has been engaged in middleware and distributed object-computing research from its concept origination with the earliest Internet to the present. He received his PhD in Computer Science from Stony Brook University in 1974 and is an ACM Fellow. richard.schantz@raytheon.com

REFERENCES

[1] Samuel Greengard. November 2019. When drones fly. *CACM,* Volume 62, No. 11, 16-18.

[2] Defense Advanced Research Projects Agency, "OFFensive Swarm-Enabled Tactics (OFFSET)." https://www.darpa.mil/work-with-us/offensive-swarm-enabled-tactics/.

[3] Georgy Skorobogatov, Cristina Barrado Muxí, and Esther Salamí San Juan. 2020. Multiple UAV systems: A survey. *Unmanned Systems* 8.2, 149-169.

[4] Timothy H. Chung, et al. 2016. Live-fly, large-scale field experimentation for large numbers of fixed-wing UAVs. *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

[5] Kevin ZY Ang, et al. 2018. High-precision multi-UAV teaming for the first outdoor night show in Singapore. *Unmanned Systems* 6.0: 39-65.

[6] CNBC, "Behind the scenes as Intel sets a world record for flying over 2,000 drones at once." https://www.cnbc.com/2018/07/17/intel-breaks-world-record-2018-drones.html

[7] Melanie Schranz, et al. April 2020. Swarm robotic behaviors and current applications, *Frontiers in Robotics and AI,* Volume 7.

[8] Michael Rubenstein, Christian Ahler, and Radhika Nagpal. 2012. Kilobot: A low cost scalable robot system for collective behaviors. *2012 IEEE International Conference on Robotics and Automation*. IEEE.

[9] Jose A. Millan-Romera, et al. 2019. ROS-MAGNA, a ROS-based framework for the definition and management of multi-UAS cooperative missions. *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2019.

[10] Jose-Luis Sanchez-Lopez, et al. 2016. Aerostack: An architecture and open-source software framework for aerial robotics. *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE.

[11] Vivek Varadharajan, et al. 2017. A software ecosystem for autonomous UAV swarms. *International Symposium on Aerial Robotics.*

[12] Soon-Jo Chung, et al. 2018. A survey on aerial swarm robotics. *IEEE Transactions on Robotics* 34.4: 837-855.

[13] Senthil Hariharan Arul, et al. 2019. LSwarm: Efficient collision avoidance for large swarms with coverage constraints in complex urban scenes. *IEEE Robotics and Automation Letters* 4.4: 3940-3947.

[14] Kourosh Naderi, Joose Rajamäki, and Perttu Hämäläinen. 2015. RT-RRT* a real-time path planning algorithm based on RRT. *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*. 2015.

[15] Nathan Koenig, and Andrew Howard. 2004. Design and use paradigms for gazebo, an open-source multi-robot simulator. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*(IEEE Cat. No. 04CH37566). Vol. 3. IEEE.

[16] Edwin Olson. 2011. AprilTag: A robust and flexible visual fiducial system. *2011 IEEE International Conference on Robotics and Automation*. IEEE.

[17] Shital Shah, et al. 2018. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. *Field and Service Robotics Conference.* Springer, Cham.

[18] P. Walker, et al. 2019. A playbook-based interface for human control of swarms. *Human Performance in Automated and Autonomous Systems: Emerging Issues and Practical Perspectives.*

[19] Raspberry Pi Foundation, 2020."Buy a Raspberry Pi 3 Model B – Raspberry Pi." https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

[20] Raspberry Pi Foundation, 2020. "Buy a Camera Module V2 – Raspberry Pi." https://www.raspberrypi.org/products/camera-module-v2/.

[21] WWingtra AG. WingtraOne – Mapping drone for high-accuracy aerial surveys. https://wingtra.com/mapping-drone-wingtraone/.

[22] Kyle Usbeck, et al. 2015. Improving situation awareness with the Android Team Awareness Kit (ATAK). International Society for Optics and Photonics. *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security, Defense, and Law Enforcement XIV*. Vol. 9456. International Society for Optics and Photonics.

[23] Pixhawk: https://pixhawk.org

[24] MAVLink: https://mavlink.io/en/

[25] Github: https://github.com/bbnsclark