SAND2015-0974C

Enhanced Physical Security through a Command-Intent Driven Multi-Agent Sensor Network

Joshua Love, Wendy Amai, Timothy Blada, Charles Little, Jason Neely, and Stephen Buerger*

Sandia National Laboratories, Intelligent Systems, Robotics and Cybernetics Group, P.O. Box 5800 MS 1010, Albuquerque, NM 87185
sbuerge@sandia.gov

Abstract. Sandia's Intelligent Systems, Robotics, and Cybernetics group (ISRC) created the Sandia Architecture for Heterogeneous Unmanned System Control (SAHUC) to demonstrate how heterogeneous multi-agent teams could be used for tactical operations including the protection of high-consequence sites. Advances in multi-agent autonomy and unmanned systems have provided revolutionary new capabilities that can be leveraged for physical security applications. SAHUC applies these capabilities to produce a command-intent driven, autonomously adapting, multi-agent mobile sensor network. This network could enhance the security of high-consequence sites; it can be quickly and intuitively re-tasked to rapidly adapt to changing security conditions.

The SAHUC architecture, GUI, autonomy layers, and implementation are explored. Results from experiments and a demonstration are also discussed.

Keywords: Command Intent · Physical Security · Heterogeneous · Multi-Agent · Distributed · Hierarchical Control · SAHUC

1 Introduction

The Intelligent Systems, Robotics, and Cybernetics group (ISRC) performs research and development in several areas of interest to Sandia [1]. One area is protecting high-consequence sites. Sandia provides cutting-edge physical-security solutions by leveraging emerging technologies. Multi-agent systems are one of these emerging technologies. The Sandia Architecture for Heterogeneous Unmanned System Control (SAHUC) was created to demonstrate how heterogeneous multi-agent systems can enhance the security of high-consequence sites. SAHUC enables a single human to easily and intuitively operate a multi-agent mobile sensor network by specifying command intent.

Recent developments in autonomy have produced sophisticated unmanned systems (UMSs). These impressive UMSs have completed complex tasks while minimally interacting with their human operators. Autonomous navigation across difficult terrain and through urban traffic has been demonstrated in [2] and [3]. These advancements are vital features of future UMS operations. However, securing an entire high-consequence facility requires a team of agents working together seamlessly, not just one highly-capable agent working alone. Advanced navigation is necessary to move individual agents around the facilities, but collaborative autonomy is also necessary to ensure the entire multi-agent team is working together toward common objectives.

If the multi-agent system's desired behavior is well-known prior to execution, scripted collaborative autonomy can sequence combinations of tasks to form complex choreographed multi-agent missions. However, this cannot flexibly respond to intelligent adversaries looking to exploit any rigidly defined and predictable behaviors [4].

Swarm autonomy enables agents to cooperate with each other on a single unifying objective through consensus methods. These methods work well when controlling large groups of simple, homogeneous agents. They are effective at tasks like plume localization, moving in formations, and spreading around perimeters [5,6]. However, physical security missions are not single objectives; they are coordinated sets of objectives.

Other collaborative autonomy approaches assign tasks to individual agents, who then execute those tasks independently with minimal coordination (typically after solving the multi-agent traveling salesman problem) [7, 8, 9]. While promising for many multi-agent applications, these approaches are not tailored to the physical security scenarios where tasks change rapidly to counter time-critical security threats.

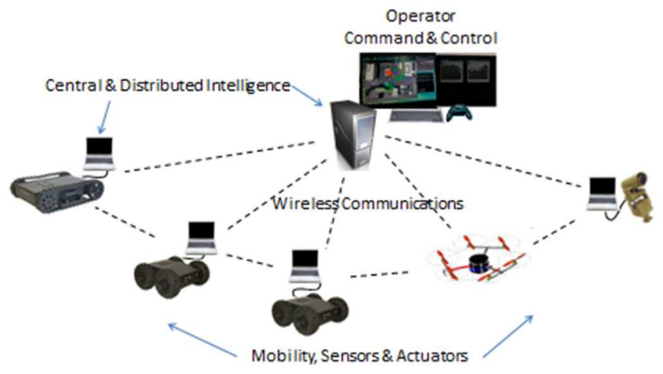
Operator
Command & Control
Central & Distributed Intelligence
Wireless Communications
Mobility, Sensors & Actuators

Fig. 1. Human operators intuitively command multi-agent sensor networks with SAHUC.

SAHUC adapts the techniques described above to produce a multi-agent system emphasizing responsiveness over optimality, figure 1. Other applications may have enough time to calculate an optimal global solution; but the physical security of high-consequence sites does not. Security threats at high-consequence sites must be dealt with immediately.

Sandia's motivation for creating SAHUC has been presented in Section 1. SAHUC's architecture is detailed in Section 2. Section 3 discusses the current SAHUC proof-of-concept implementation. Section 4 presents the results of two experiments quantifying SAHUC's performance. Section 5 recounts a physical security demonstration, illustrating the potential usefulness of SAHUC. Finally, Section 6 concludes.

2 SAHUC Architecture

SAHUC enables a single human to operate a multi-agent mobile network, figure 1. This requires several levels of hierarchy, figure 2. The Graphical User Interface (GUI) allows the operator to specify command intent by adding, removing, or modifying objectives for the network to accomplish. The human can also specify how the network is expected to perform those objectives by modifying weighting parameters. The high-level autonomy then assigns the objectives to agents. The mid-level autonomy plans paths for the assigned objectives. The mid-level autonomy also estimates metrics characterizing the agent's expected performance. These metric estimates are combined, along with the weighting parameters, to form the costs used in the high-level assignment algorithm. Finally, the lowest level of autonomy tracks the planned path and estimates the agent's state.

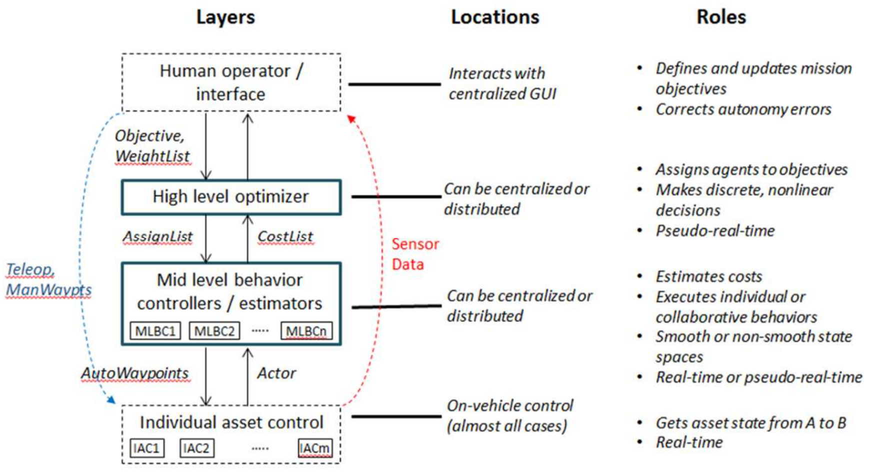
Layers
Locations
Roles
Human operator / interface
Interacts with centralized GUI
Defines and updates mission objectives
Corrects autonomy errors
Objective, WeightList
High level optimizer
Can be centralized or distributed
Assigns agents to objectives
Makes discrete, nonlinear decisions
Pseudo-real-time
AssignList
CostList
Sensor Data
Teleop, ManWaypts
Mid level behavior controllers / estimators
MLBC1
MLBC2
MLBCn
Can be centralized or distributed
Estimates costs
Executes individual or collaborative behaviors
Smooth or non-smooth state spaces
Real-time or pseudo-real-time
AutoWaypoints
Actor
Individual asset control
IAC1
IAC2
IACm
On-vehicle control (almost all cases)
Gets asset state from A to B
Real-time

Fig. 2. A conceptual view of SAHUC's hierarchy

2.1 Graphical User Interface

The GUI provides an intuitive 3D view of the current state of the network, figure 3. It shows where the agents and adversarial targets currently are positioned and where they have recently been. By right-clicking in the 3D display a user can quickly create a point, multi-segment line, or area of interest. By right-clicking on that point/line/area a new objective can be generated (e.g. look at the point, patrol the border, or search the area). A similarly trivial procedure can be applied to any agent/target icon to create a new objective (e.g. to escort the agent, or follow the target). Through this method, the operator constructs high-level mission behaviors by specifying desired outcomes. If the operator prefers to bypass the autonomy, they can directly command the agents by manually specifying a waypoint or manually teleoperating the vehicle with a joystick. The GUI also displays live sensor streams.

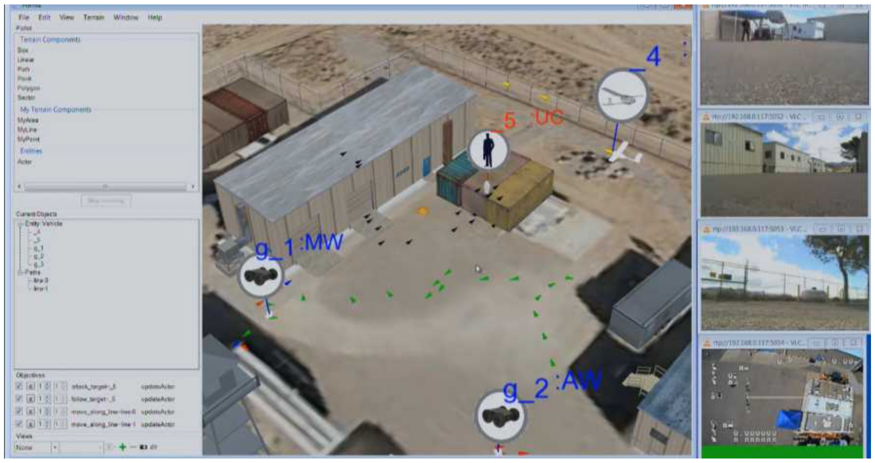

Fig. 3. SAHUC's Graphical User Interface

2.2 High-Level Autonomy

The highest level of autonomy contains an assignment algorithm. It automatically assigns objectives to agents using metrics computed by the mid-level autonomy. While the assignment algorithm is based upon optimization techniques, approximations are used to speed up the algorithm. Responsiveness and adaptability to changing security threats is more important that being truly "optimal". When a new threat is identified, new objectives are created. The system must respond in seconds; not stop, fully re-plan a truly optimal solution, and then act minutes later.

To ensure assignments occur quickly, metrics are computed in a distributed manner onboard the agents. They are coarse estimations of the true values. The metrics are communicated and combined to form the costs used in the assignment algorithm.

$$C(A_i, O_j) = \beta_1 \cdot M_1(A_i, O_j) + \beta_2 \cdot M_2(A_i, O_j) + \cdots \beta_n \cdot M_n(A_i, O_j) \tag{1}$$

The estimated cost of agent A_i performing objective O_j is a weighted combination of the metrics calculated by A_i, equation (1). β_k are the weighting parameters defined through the GUI. They enable the human operator to modify how the agents operate (e.g. switch from "fast" to "energy efficient" by modifying the metric weights).

The assignment algorithm minimizes the cost J defined by:

$$J = \sum_i^{N_A} \sum_j^{N_O} z_{ij} \cdot C(A_i, O_j) \tag{2}$$

N_A and N_O are the number of agents and objectives. z_{ij} are a set of binary assignment variables. If agent A_i determined that executing objective O_j was infeasible, a constraint is added to the optimization to prevent this assignment from occurring: $z_{ij}=0$. Typically, a 1:1 assignment is assumed. That is, each agent is assigned to at most one objective, and one objective has at most one agent assigned to it. In this case, the constraints on equation (2) are:

$$\sum_i^{N_A} z_{ij} \leq 1 \forall j, \quad \sum_j^{N_O} z_{ij} \leq 1 \forall i, \quad \sum_i^{N_A} \sum_j^{N_O} z_{ij} = \min(N_A, N_O) \tag{3}$$

The third constraint prevents agents from being lazy by forcing as many assignments to occur as possible. By allowing the binary decision variable z to be $z \epsilon [0,1]$, conventional linear programming methods can be applied [10]. These conventional methods can decrease the computation time required for a solution.

In order to prevent the assignment algorithm from suffering "decision chatter" (excessive switching between assignments), a switching cost is assigned to changing from the current assignment, adding hysteresis [11, 12].

If multiple agents are to be assigned to a single objective (e.g. to enable swarm-based objectives), the constraints in (3) must be modified to be:

$$\sum_i^{N_A} z_{ij} \leq maxsize\left(\mathrm{O_j}\right), \sum_j^{N_O} z_{ij} \leq 1 \forall i, \sum_i^{N_A} \sum_j^{N_O} z_{ij} = \min\left(N_A, \sum_j^{N_O} maxsize\left(O_j\right)\right)$$

Maxsize is the maximum number of agents that can be assigned to each objective.

2.3 Mid-Level Autonomy

The mid-level autonomy plans a path to execute the agent's assigned objective. It also considers alternative objectives by: determining if the alternative is feasible, planning a path for the alternative, and estimating metrics for the alternative. These metric estimates are communicated to the high-level optimization layer discussed above.

The mid-level behaviors can determine paths for individuals or teams/swarms. For Physical Security it is necessary to have agents perform behaviors like: move to a point, search an area, patrol a border, protect another asset, follow a target, etc. Heterogeneity in the agent's dynamics or computational capabilities may require different path planning algorithms for the same type of objective. Consider moving to a specific point: the path for a UAV to fly over a building is radically different from the path for a UGV to go around it. The algorithms producing the paths must simply agree on the inputs necessary to define the point to which they are moving.

The mid-level autonomy also produces metric estimates for all objectives. First, feasibility is determined. An objective may be infeasible for an agent because: it may lack a path planning algorithm for that type of objective, it may not have the necessary sensor suite, its limited resources may not allow it to complete the objective.

If the agent could feasibly execute the objective, a path is planned using the mid-level behavior's algorithm. The fidelity of the path is not constrained within SAHUC. Computationally limited agents can use very coarse approximations while computationally rich agents can use highly-accurate predictions. This allows all agents to rapidly approximate the paths for all alternative objectives. The metrics are then calculated for each alternative objective. These metrics include transient and steady-state effects: time to arrival, cycle time, energy to arrival, cycle energy, distance to arrival, cycle distance. For example, the border patrol in figure 4 has a distance to arrive at the border (segment a), and then a different distance to perform one complete cycle of that patrol (segments b through g).

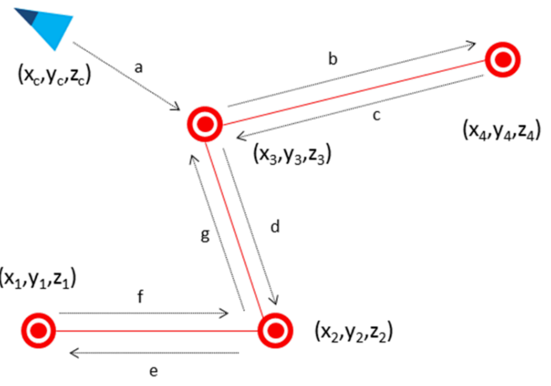

Fig. 4. An example of a border patrol, segments b-through-g would repeat indefinitely.

2.4 Low-Level Autonomy

The low-level autonomy follows the planned paths and produces state estimates. The algorithms chosen also depend on the computation available. Heterogeneous agents have heterogeneous capabilities and dynamics. SAHUC does not mandate which algorithms are used in the low-level autonomy.

SAHUC's control algorithms can vary from PIDs to Model-Predictive Controllers to Rapidly-expanding Random Trees. Vehicle-specific controllers are used to fully exploit knowledge of the agent's dynamics (e.g. skid-steered vs unicycle UGVs).

Similarly, the state estimation algorithms vary based on sensor suite and computational capability. Computationally limited agents can use simple Kalman Filters with GPS and IMU measurements. More capable agents can use full SLAM solutions. If the state estimation algorithms include local environment representations, obstacle avoidance can be incorporated within the low-level autonomy.

3 Proof-of-Concept Implementation

SAHUC's proof-of-concept implementation leveraged Commercial-Off-The-Shelf hardware and open-source software. Three highly-modified Jaguar UGVs from Dr. Robot are shown in figure 5 [13]. Dr. Robot's GPS receivers were replaced with better-performing NovAtel receivers and pinwheel antennas. The pinwheel antennas and the Jaguar's IMUs where moved outside of the main vehicle bodies to reduce EMF and multi-path interference from the motors and the ground. Asus netbooks with a 1.6 GHz Atom processors running Ubuntu 12.04 were added to each vehicle. Webcams and Hokuyo URG-04LX lidars were also added.

Figure 5 shows three 3DR Arducopter UAVs [14]. The Arducopters have onboard GPS/IMU, 2.4 GHz RC control links, 1.3 GHz video links, and 900 MHz base station links. The UAV base stations operated on 1.6 GHz Asus netbooks. This allowed the UAVs to have identical processing capabilities as the UGVs, but the processing was located offboard due to the small UAVs' limited payloads.

Additionally, GPS backpacks simulated computer vision. The GUI was run on a Windows 7 laptop. All wireless communications used standard 802.11g hardware.

The user interface was implemented using Sandia's UMBRA framework [15]. UMBRA provides a 3D model-based interface that was extended for the SAHUC implementation. An XBOX 360 controller enabled teleoperation through the GUI.

The high-level autonomy's assignment algorithm is de-centralizable, but was implemented in a centralized fashion using Matlab's Optimization toolbox, primarily due to time constraints. The mid-level autonomy was implemented in a distributed fashion using the Robot Operating System (ROS) [16]. The mid-level behaviors and metric estimators were custom-written ROS processes. The information exchanged with the high-level autonomy used UDP datagrams. The low-level autonomy was also implemented in a distributed fashion using ROS. Custom Extended Kalman Filters were written for state estimation. Simple PID controllers closed turn-rate and velocity loops for waypoint tracking. These two simple algorithms were computationally efficient alternatives to ROS's slam_gmapping and move_base packages. Those packages were also integrated to enable SLAM and advanced path following (including obstacle avoidance). However, they taxed the netbooks to their computational limits and the lidars needed for SLAM did not function well outdoors during daylight hours.

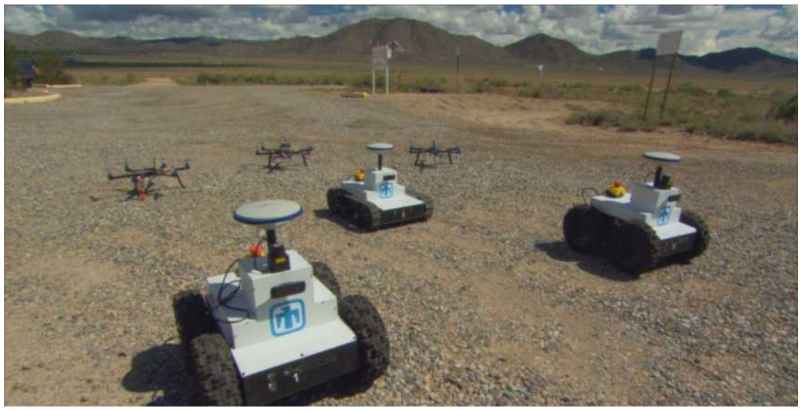

Fig. 5. UGVs and UAVs for SAHUC's proof-of-concept implementation

4 Experiments

Two simple experiments were performed to verify SAHUC's anticipated performance. Each experiment was performed ten times to gather sufficient statistical data.

4.1 Experiment 1

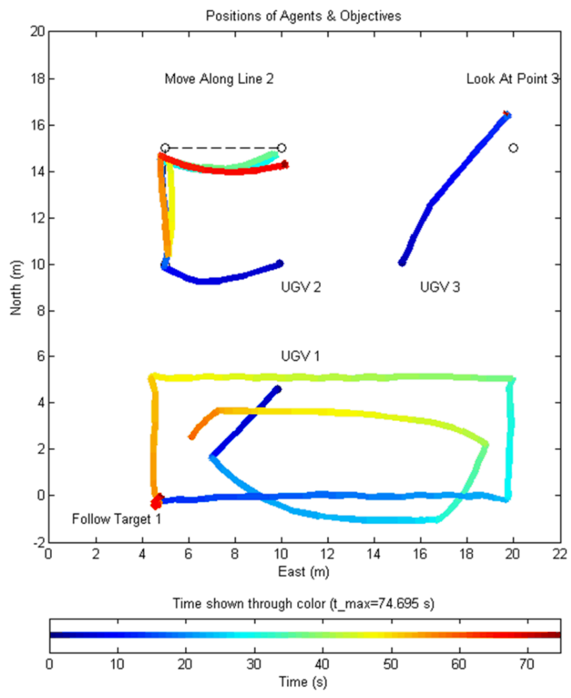

Fig. 6. Experiment 1 had three UGVs performing three objectives: follow a target, move along a line, and look at a point from the North. Time is indicated through color.

The first experiment, shown in figure 6, had three UGVs and three objectives: follow a target (that walked in a predefined box pattern), move along a 2-segment line, and look at a point from the North. The assignment algorithm was set to run every 0.1 seconds and the measured rate was 0.101 seconds (std dev 0.046). The slight drop and variability can be attributed to running Matlab on a non-real time operating system (Windows 7). The agents were set to compute and broadcast their information every 0.5 seconds; the measured rate was 0.506 seconds (std dev 0.077). Experiment 1 verified that SAHUC's proof-of-concept implementation can calculate the metrics and the assignment algorithm's solution as quickly as expected. It also confirmed that the assignment algorithm produced the obviously correct assignments.

4.2 Experiment 2

The second experiment, shown in figure 7, had two identical UGVs and two objectives: look at a point from any direction, and follow a target. At the beginning of the experiment, UGV 1 was best suited to follow the target (because it was closest and requires the shortest time, distance, and energy). UGV 2 was similarly best suited to look at the point. Figure 7's initial trajectories demonstrate that this is what the assignment was, until the target made a short movement to quickly alter the obvious best assignment. The UGV's can be seen to quickly switch objectives to automatically adapt to the changing situation (the switch would happen even faster if the switching costs adding hysteresis were removed). As the target moved, each UGV's metrics for following the target changed. These metrics were quickly computed and communicated to the assignment algorithm. The assignment algorithm quickly determined that it was best to switch the assignment. Finally, the agents quickly changed course. The entire end-to-end process takes less than a few seconds, verifying that SAHUC's proof-of-concept implementation is highly responsive and can rapidly and autonomously adapt to changing situations.

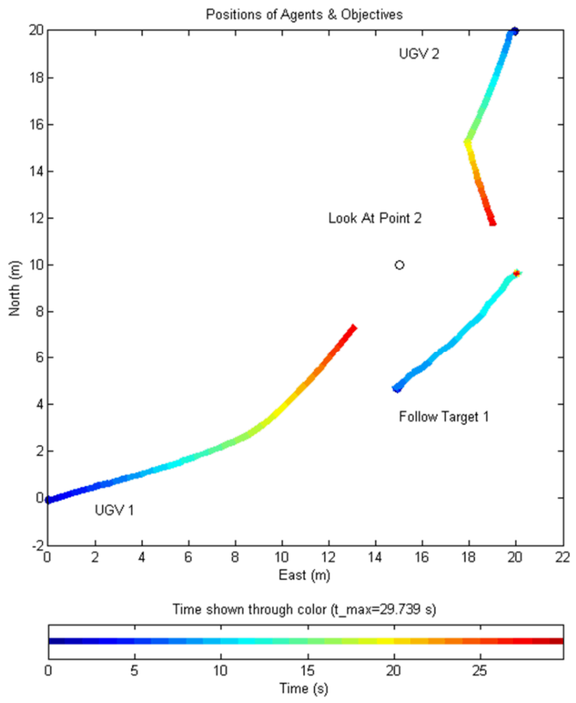

Fig. 7. Experiment 2 had two UGVs and two objectives: look at a point from any direction, follow a target. The target made a short quick move, causing a change in the assignment.

5 Physical Security Demonstration

SAHUC's physical security potential was demonstrated at Sandia's Robotic Vehicle Range (RVR). The RVR includes buildings, towers, paved areas, gravel areas, dirt paths, and a fenced perimeter; making it an ideal physical security test site, figure 8.

All three UGVs were used in the demonstration, but only one UAV was flown, in order to satisfy safety concerns. The UAV had a safety pilot to take over RC control in the event of an emergency. The UAV also had a FAA licensed private pilot-in-command to monitor the UAV, communicate with the control tower, and monitor the sky.

The pilot-in-command and safety pilot were not active participants in the system, just necessary to ensure safe testing. The human operator ran the GUI from inside a control room in the RVR, having no line-of-sight to the agents being commanded. The agents operated inside and outside of the fenced security perimeter. Finally, an intruder approached the RVR from the Southeast, sneaked through an open gate on the Southeast end of the RVR, and attempted to gather intelligence about the facility.

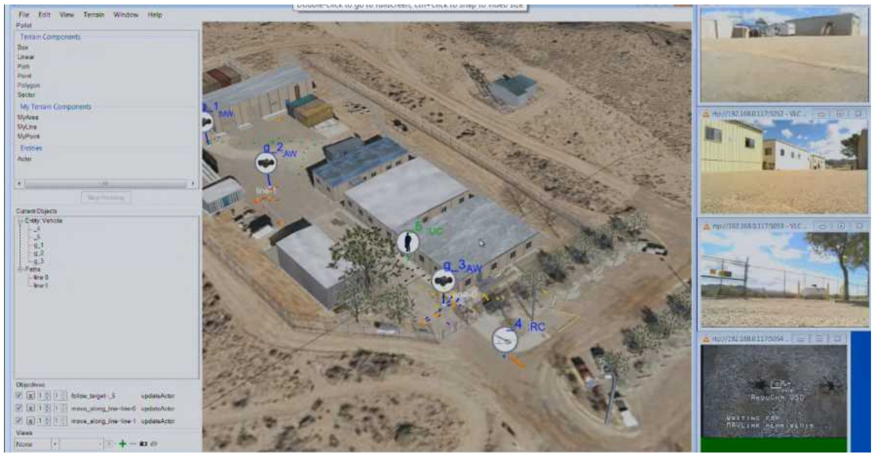

Fig. 8. GUI screenshot right after the intruder passed through the open gate.

The human operator began by distributing the UGVs throughout the facility. A border patrol objective was created at the open gate because the operator knew this was a weak point in the current security. The tracked UGV began to immediately move back and forth along the open gate, streaming video back to the operator. Then, the operator manually spread the other two UGVs throughout the facility. A visit point objective and an additional border patrol objective were created to place the other UGVs at strategic locations (near the corners of buildings) in the RVR facility.

At this point, the intruder sneaked through the open gate behind the patrolling UGV, but was detected with the camera once the UGV turned around upon reaching the North end of the gate. In practice, machine vision would be used to detect the intruder; however, due to time constraints this was simulated through a GPS back-pack. Figure 8 is just after this moment.

Upon the detection of an intruder, the human operator created a track target objective. This was automatically assigned to the UAV, because the UAV was significantly faster at getting to and continuing to track the person. The UAV continued to track the intruder until the intruder disappeared from sight by moving under a tarp, figure 9.

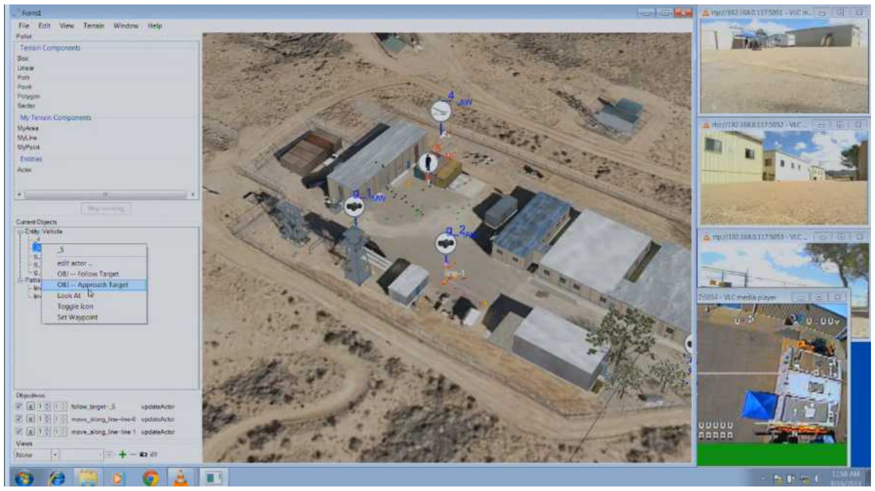

Fig. 9. GUI screenshot when the intruder is hiding under the blue tarp.

The human operator created a new look-at-point objective to get streaming video of what was under the blue tarp. The closest UGV was assigned and provided video of the intruder. Finally, the human operator informed the intruder that they had been detected and should surrender immediately using speakers on the UGV.

The demonstration illustrated how multiple agents could enhance the physical security of a high-consequence site. The need for heterogeneity was shown when the UAV could better track the intruder in the open, but the UGV could easily see under the blue tarp. The impact of multi-agent autonomy was clear; the single human operator controlled all 4 UMSs simultaneously, and monitored the live sensor feeds, while also explaining the system and answering questions to demonstration attendees.

6 Conclusions

SAHUC enables multi-agent mobile sensor networks for securing high-consequence sites. Multi-agent algorithms have been adapted to emphasize timely responsiveness over optimality, producing a system that can adapt in seconds to constantly evolving security conditions. The capability leverages autonomy where possible and enables operator attention to be focused on the highest-level tasks, such as tactical reasoning.

As processors improve, more computationally expensive algorithms (e.g. SLAM and obstacle avoidance) can be run on the agents. These algorithms should continue to maintain SAHUC's responsiveness. The distributed computation of metrics is a major strength of the architecture. The ability to plan paths and estimate metrics at varying levels of coarseness further ensure responsiveness by allowing customization to the individual agent's computational capability. SAHUC's major drawback is its dependence upon communications, but the bandwidth required for SAHUC's information is significantly lower than that of streaming the live video feeds.

Future work includes larger scale demonstrations, potentially including of all three UAVs, which will yield more complete and realistic demonstrations. The video feeds should be directly overlaid onto the 3D map, making it more intuitive to see what the agents are seeing. Real computer vision algorithms should be run onboard the agents to replace the GPS backpacks. These algorithms may require the increasing the onboard computing. Finally, communications should be intentionally degraded to evaluate how the multi-agent network performs while operating under interference.

Acknowledgments. This research was funded by the Laboratory Directed Research and Development (LDRD) program at Sandia National Laboratories. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

References

1. Intelligent Systems, Robotics, & Cybernetics, `http://www.sandia.gov/research/robotics/index.html`
2. Urmson, C. et al., "A Robust Approach to High-Speed Navigation for Unrehearsed Desert Terrain", J. Field Rob. 23(8), 467-508 (2006)
3. Montemerlo, M. et al., "Junior: The Stanford Entry in the Urban Challenge", J. Field Rob. 25(9), 569-97 (2008)
4. Kira, Z. et al., "A Design Process for Robot Capabilities and Missions Applied to Micro-Autonomous Platforms", Proc. SPIE 7679, 767911 (2010)
5. Vincent, R. et al, "Distributed Multirobot Exploration, Mapping, and Task Allocation", Ann. Math. Artif. Intell. 52(2), 229-255 (2009)
6. Scardovi, L. et al., "Stabilization of Three-Dimensional Collective Motion", Comm. Inf. Sys. 8(4), 473-500 (2008)
7. Ryan, A. et al., "Decentralized Control of Unmanned Aerial Vehicle Sensing Missions", Proc. IEEE Amer. Contr. Conf., 4672-7 (2007)
8. Garvey, J. et al, "An Autonomous Unmanned Aerial Vehicle System for Sensing and Tracking." Infotech@ Aerospace Conference (2011)
9. Bertucelli, L. F. et al., "Robust Planning for Coupled Cooperative UAV Missions", Proc. IEEE Conf. Dec. Contr., 2917-2922 (2004)
10. Linear Programming Relaxation, Wikipedia, `http://en.wikipedia.org/wiki/Linear_programming_relaxation`
11. Dusonchet, F. and Hongler, M. O., "Optimal Hysteresis for a Class of Deterministic Deteriorating Two-armed Bandit Problem with Switching Costs", Automatica 39(11), 1947-55 (2003)
12. Kitaev, A., Yu., M., Serfozo, A., and Richard F., "M/M/1 Queues with Switching Costs and Hysteretic Optimal Control", J. Operations Research 47(2), 310-2 (1999)
13. `http://jaguar.drrobot.com`
14. `http://3drobotics.com`
15. `http://umbra.sandia.gov`
16. `http://www.ros.org`