SAND2015-2097C

The Sandia Architecture for Heterogeneous Unmanned System Control (SAHUC)

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

ABSTRACT

The Sandia Architecture for Heterogeneous Unmanned System Control (SAHUC) was produced as part of a three year internally funded project performed by Sandia's Intelligent Systems, Robotics, and Cybernetics group (ISRC). ISRC created SAHUC to demonstrate how teams of Unmanned Systems (UMS) can be used for small-unit tactical operations incorporated into the protection of high-consequence sites. Advances in Unmanned Systems have provided crucial autonomy capabilities that can be leveraged and adapted to physical security applications. SAHUC applies these capabilities to provide a distributed ISR network for site security. This network can be rapidly re-tasked to respond to changing security conditions.

The SAHUC architecture contains multiple levels of control. At the highest level a human operator inputs objectives for the network to accomplish. The heterogeneous unmanned systems automatically decide who can perform which objectives and then decide the best global assignment. The assignment algorithm is based upon coarse metrics that can be produced quickly. Responsiveness was deemed more crucial than optimality for responding to time-critical physical security threats. Lower levels of control take the assigned objective, perform online path planning, execute the desired plan, and stream data (LIDAR, video, GPS) back for display on the user interface. SAHUC also retains an override capability, allowing the human operator to modify all autonomous decisions whenever necessary.

SAHUC has been implemented and tested with UAVs, UGVs, and GPS-tagged blue/red force actors. The final demonstration illustrated how a small fleet, commanded by a remote human operator, could aid in securing a facility and responding to an intruder

Keywords: SAHUC, physical security, heterogeneous, distributed, unmanned, ISR, architecture, hierarchical control

1. INTRODUCTION

One of the core missions of the Intelligent Systems, Robotics, and Cybernetics group (ISRC) at Sandia National Laboratories is to develop advanced technology for the protection of high-consequence sites. To provide cutting-edge physical security solutions, Sandia investigates a variety of relevant emerging technologies. Unmanned Systems (UMS) and autonomy are two prominent and highly relevant technologies. ISRC developed the Sandia Architecture for Heterogeneous Unmanned System Control (SAHUC) to demonstrate how these new capabilities could be leveraged to augment the security of high-consequence sites.

Recently, highly capable autonomy has been embedded on individual UMS platforms. This has allowed UMSs to complete complex tasks with humans providing only minimal high-level oversight. Most visibly, autonomous navigation has been demonstrated across difficult terrain[2] and through urban traffic[3]. These capabilities represent important elements for future UMS operations. However, high-consequence physical security applications require teams of UMS to secure an entire facility, not just one portion of it. Advanced navigation autonomy is necessary to move individual team members safely and quickly around facilities, but collaborative autonomy is also necessary to ensure the team members are efficiently working together toward common objectives.

*sbuerge@sandia.gov; phone (505)284-3381; fax (505)844-8323; www.sandia.gov

+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.

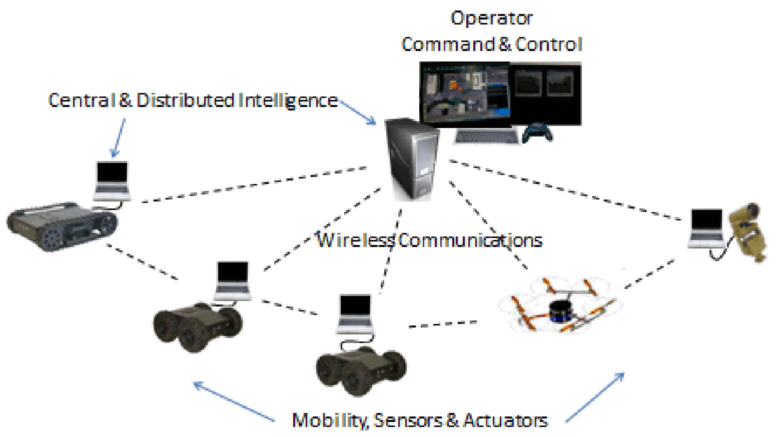

Operator
Command & Control
Central & Distributed Intelligence
Wireless Communications
Mobility, Sensors & Actuators

Figure 1. SAHUC allows a single human operator to intuitively command a distributed team of heterogeneous UMS.

Scripted collaborative autonomy techniques, wherein missions are built up from pre-sequenced combinations of tasks, can be effective in well-known and predictable conditions. However, these techniques are not conducive to quickly responding to intelligent adversaries seeking to exploit any rigidly defined and predictable behaviors[4].

Swarm autonomy techniques enable multiple assets to work together on a single unifying objective using consensus methods. These approaches perform best when controlling tens to hundreds of very simple homogeneous UMS and are effective at tasks like plume localization, moving in formation, and spreading around perimeters[5,6]. However, relatively few physical security missions can be entirely distilled to this sort of single objective. Additionally, these behaviors fail to take advantage of the multi-functional capabilities provided by modern military-grade UMS assets.

Other approaches to collaborative autonomy optimize the assignment of tasks (e.g. typically visiting fixed waypoints) to individual UMSs, which then execute those tasks independently with minimal coordination[7-9]. While promising for many types of operations, these methods are not designed for the physical security scenarios where tasking may change extremely rapidly in response to time-critical security threats.

The Sandia Architecture for Heterogeneous Unmanned System Control adapts collaborative autonomy techniques to produce a system that emphasizes timely responsiveness over full optimality, figure 1. While other applications may have the luxury of calculating a truly optimal global solution offline, the physical security of high-consequence sites is not compatible with this approach. When security threats are encountered, they must be dealt with swiftly; the system must be highly reactive to changing conditions. The trade-off between optimality and computing resources has been well recognized in the context of optimal control. In Scokaert, Mayne, and Rawlings[10], the authors remark that many "[s]tabilizing formulations … rely on the assumption that global and exact solutions of nonconvex, nonlinear optimization problems are possible in limited computational time" and that this is in general not practical. Instead, the authors identify stability to be the priority when computational resources are limited and establish a method wherein suboptimal control policies may be generated while stability is still guaranteed. In Zeilinger, Jones, and Morari[11], the authors use offline (presolved) optimization results to provide a "warm-start" to a real-time optimization algorithm, reducing the number of iterations required for solution. In this case, the offline solution provides a "good enough" solution, but the solution may be refined further is computational resources permit.

Section 1 has briefly discussed Sandia's motivation for creating SAHUC. Section 2 will detail the contents of the SAHUC architecture. The proof-of-concept implementation is described in Section 3. Section 4 presents a physical security demonstration performed at Sandia to illustrate the potential utility of SAHUC. Finally, Section 5 contains conclusions, an assessment of the strengths and weaknesses of the approach, and some directions for future work.

2. SAHUC ARCHITECTURE

The SAHUC architecture allows a single human operator to command a distributed network of collaborating UMSs, figure 1. To accomplish this, several levels of autonomy are connected hierarchically. Figure 2 depicts this hierarchy conceptually. The lowest level of autonomy contains the traditional path tracking control as well as state estimation. The mid-level autonomy contains a set of behavior algorithms to generate the paths needed by the low-level layer (e.g. it can plan how to search an area or track a moving target). These mid-level behaviors can be for individual UMSs or swarms of UMSs working together on a single objective. The mid-level layer also estimates several metrics to characterize the expected performance of the UMSs if they executed the objectives. The high-level optimizer contains an optimization-based assignment algorithm to assign the UMSs to objectives, based upon costs formed from the metric estimates produced by the mid-level autonomy. Finally, the Graphical User Interface (GUI) allows the human operator to add, remove, or modify the objectives in the system. They can also change how the metrics are combined to form the costs being minimized (e.g. the human operator can move from desiring “fastest” to “most energy efficient”).

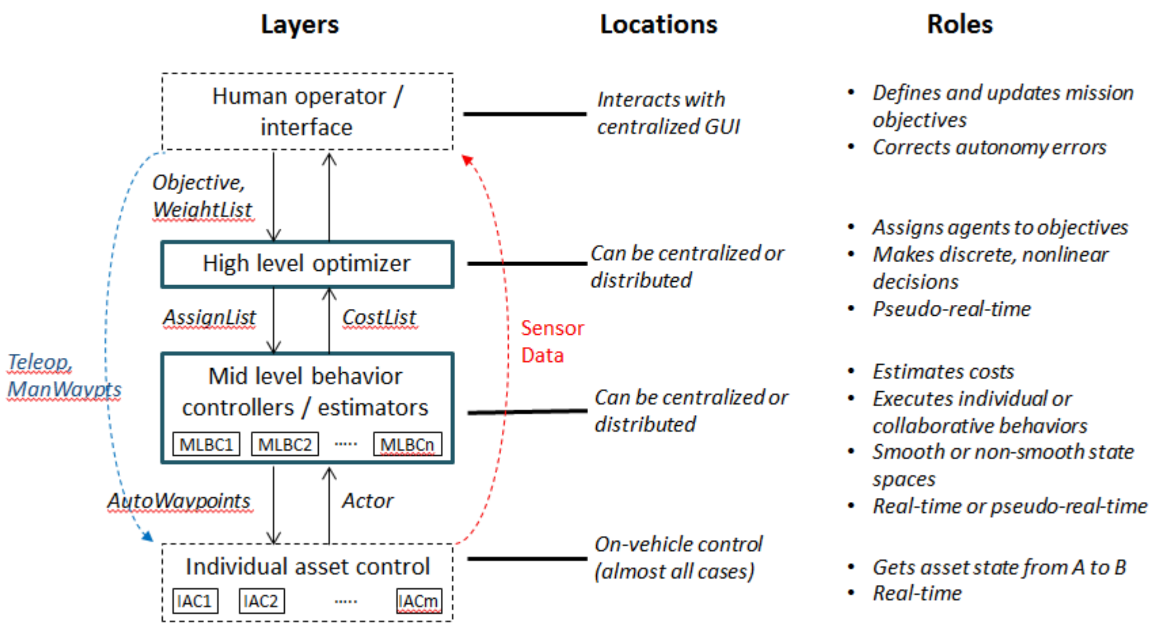

Layers
Locations
Roles
Human operator / interface
Interacts with centralized GUI
Defines and updates mission objectives
Corrects autonomy errors
Objective, WeightList
High level optimizer
Can be centralized or distributed
Assigns agents to objectives
Makes discrete, nonlinear decisions
Pseudo-real-time
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

Figure 2. Conceptual hierarchy of the Sandia Architecture for Heterogeneous Unmanned System Control.

2.1 Low-level Autonomy

The lowest level of autonomy accepts desired states or paths to follow and produces state estimates for the UMS. This is generally performed onboard the UMS, but could be performed by a surrogate (e.g. a ground station or neighboring vehicle). The specific algorithms chosen depend on the computational resources available. Different UMS may have different capabilities and different vehicle dynamics. SAHUC does not specify or constrain what algorithms are used wherever possible, including within the low-level autonomy.

SAHUC allows a wide variety of low level control algorithms including PIDs, Model-Predictive Controllers, Rapidly-expanding Random Trees and other approaches. It also allows for vehicle specific controllers to be used to fully exploit knowledge of the individual UMSs dynamics (e.g. skid-steered vs unicycle UGVs).

Similarly, the state estimation algorithms can vary based on sensor suite and computational resources. Simple Extended Kalman Filters or Unscented Kalman Filters can be used with GPS and IMU measurements for computationally limited UMSs, but full SLAM solutions can be used for other more advanced platforms. If the state estimation algorithms include representations of the local environment, local obstacle avoidance may also be incorporated within the low-level autonomy.

2.2 Mid-level Autonomy

The mid-level autonomy executes the UMS's assigned objective by planning waypoints or paths that are then sent to the low-level autonomy discussed above. It also evaluates every other objective, determines if the objective is feasible, plans a tentative path if it was to switch to that objective, and estimates the metrics associated with each of those alternative objectives. These metrics are then sent to the high-level optimization layer.

The mid-level behaviors are domain specific and can potentially determine paths for individuals or teams. For Physical Security it is necessary to have UMSs perform behaviors like: move to a point, search an area, patrol a border, protect another asset, follow a target, etc. Just as different low-level algorithms can be used, different path planning algorithms can exist for the same type of objective. This may be necessary due to heterogeneity in the UMS dynamics or the computational resources available. For example, the path for a UAV to search an area can be radically different from the path for a UGV; and the algorithms that produce the paths can be equally unique and customized. These algorithms must simply agree on the inputs necessary to define the area to be searched.

The mid-level autonomy also contains a metrics estimator, producing metric estimates for all objectives. The first portion of the estimate is to determine feasibility. An objective may not be feasible for a UMS because the UMS does not have a path planning algorithm for that type of objective. The UMS may not have the necessary sensor suite. It may also be infeasible for the UMS because the UMS's constrained resources will not allow it to reach and complete the objective (e.g. the battery will die before the UAV reaches the area to be searched).

If the UMS could feasibly execute the objective, a path is planned using the mid-level behavior's algorithm. Again, the fidelity of the path is not constrained within SAHUC. Computationally limited UMSs can use very coarse approximations while computationally rich UMSs can use detailed highly-accurate simulations. The idea is to allow each UMS to rapidly approximate the paths for each alternative objective. Multiple metrics are then calculated for each of the alternative courses of action. These metrics include measurements of transient and steady-state effects: time to arrival, cycle time, energy to arrival, cycle energy, distance to arrival, cycle distance. For example, to patrol a border a certain amount of time is required to arrive at the border, and then a different amount of time is required to perform one complete cycle of that patrol.

2.3 High-level Autonomy

The highest level of autonomy takes the objectives, the UMSs, and the metrics provided by the UMSs, and determines which objective each UMS should perform. This assignment algorithm is intended to solve a high-level multi-objective optimization, but multiple approximations are used to speed up the algorithm. The responsiveness of the system to changes in the security threat conditions is much more important that being fully optimal. Once a new threat is identified and new objectives created, the system must respond in seconds; not stop, fully re-plan an optimal solution, and then act minutes later.

To speed up the assignment, the metrics are computed in a distributed manner onboard the UMSs and are only coarse approximations of their true values. The estimated metrics are then communicated and combined to form the costs used in the assignment algorithm.

$$C(A_i,O_j)=\beta_1 M_1(A_i,O_j)+\beta_2 M_2(A_i,O_j)+\cdots+\beta_n M_n(A_i,O_j) \qquad (1)$$

Equation (1) expresses that the estimated cost of the i-th agent (A_i) performing the j-th objective (O_j) is a weighted combination of the metric values calculated by A_i. β_k are weights, defined through the graphical user interface. This gives the human operator the ability to modify how the UMS network is operating during execution (e.g. to switch from "fast" to "energy efficient").

The assignment algorithm minimizes the cost *J* defined by:

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

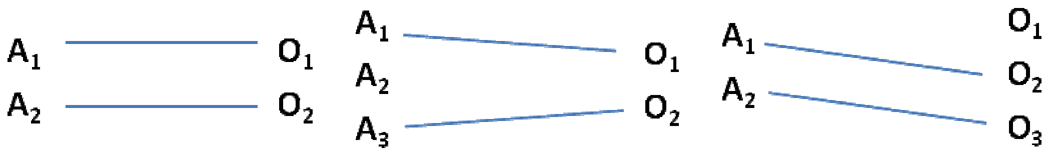


Figure 1. 1:1 assignment of agents to objectives when $N_A = N_O$, $N_A > N_O$, and $N_A < N_O$.

N_A and N_O are the number of agents and objectives, respectively, $\hat{C}(A_i, O_j)$ is the estimated cost of the i-th agent completing the j-th objective, and 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: $z_{ij}=0$. The most straightforward solution to minimize equation (2) makes 1:1 assignments of agents to objectives. 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:

a) $\sum_i^{N_A} z_{ij} \leq 1 \forall j$ (each objective is assigned to at most one agent)

b) $\sum_j^{N_O} z_{ij} \leq 1 \forall i$ (each agent is assigned to at most one objective)

c) $\sum_i^{N_A} \sum_j^{N_O} z_{ij} = \min(N_A, N_O)$ (forces as many assignments as the lesser of the number of agents and the number of objectives)

By *relaxing* the binary decision variable such that $z \in [0,1]$, conventional linear programming methods can be applied[10]. These conventional methods can decrease the computation time required to find a solution. In practice, it is observed that as long as the sensitivity of J with respect to each z_{ij} is nonzero, z_{ij} tends to be driven to the constraint limits 0 or 1, typically providing the desired assignments. If $N_A = N_O$, then each agent is assigned to an objective and each objective has an agent assigned to it. If $N_A > N_O$, then N_O assignments are made and the remaining agents remain idle. If $N_A < N_O$, then N_A assignments are made and some objectives are unassigned. These cases are shown in
Figure **1**.

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. This switching cost (hysteresis) provides some resistance to changing the assignment[13,14].

If multiple agents are allowed to be assigned to a single objective (for executing a swarm-based objective), constraint a) must be replaced with:

a$^+$) $\sum_i^{N_A} z_{ij} \leq maxteamsize(O_j)$

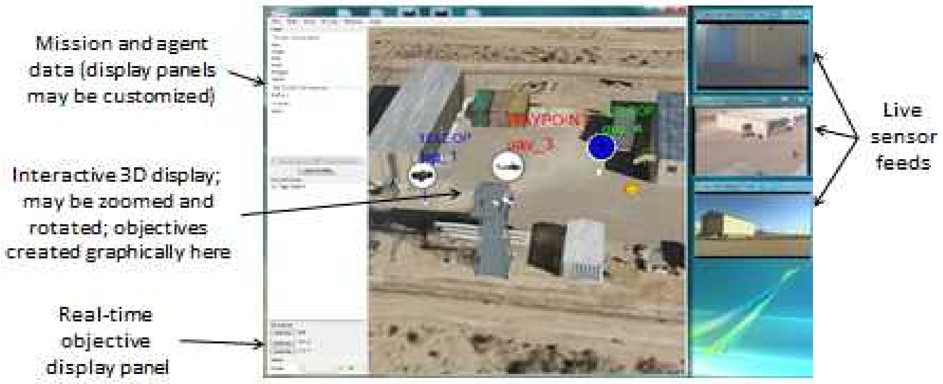
and constraint c) must be replaced with

c$^+$) $\sum_i^{N_A} \sum_j^{N_O} z_{ij} = \min\left(N_A, \sum_j^{N_O} maxteamsize(O_j)\right)$

where *maxteamsize* returns the maximum allowable number of agents that can be assigned to each objective.

2.4 User Interface

The graphical user interface (GUI), figure 4, allows the human operator to add, remove, and modify the objectives currently in the system; view the current state of the system (locations of all UMS); view the current live sensor feeds; adjust the weighting factors affecting system performance; and override every level of autonomy. High-level objectives may be specified by interacting with icons on the 3D display or with lists of agents in the simulation. The human operator can force specific assignments, manually specify waypoints, and even teleoperate individual vehicles whenever necessary. The GUI includes a 3D model representation of the operating area to provide context for live sensor feeds, which are also presented.


Mission and agent data (display panels may be customized)
Interactive 3D display; may be zoomed and rotated; objectives created graphically here
Real-time objective display panel
Live sensor feeds
Figure 4. Graphical User Interface.

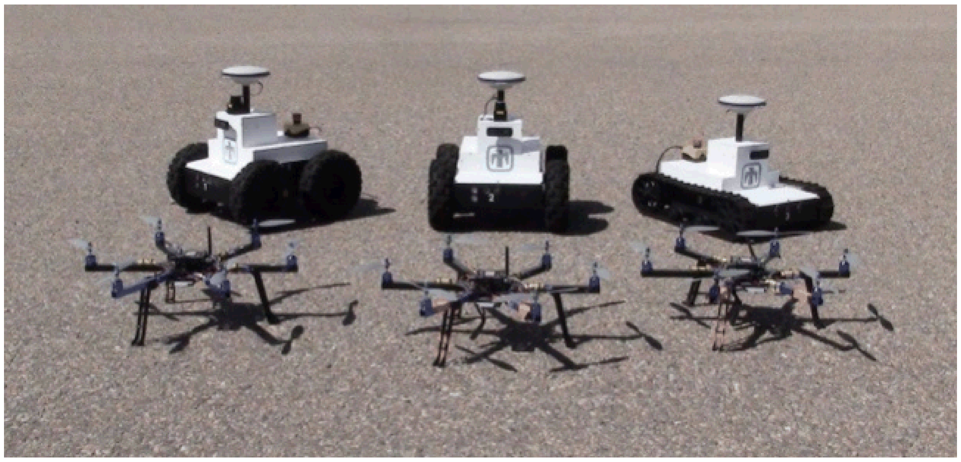


Figure 5. Modified COTS UAVs and UGVs used in SAHUC's proof-of-concept implementation.

3. IMPLEMENTATION

The proof-of-concept SAHUC implementation leveraged Commercial-Off-The-Shelf (COTS) hardware and open-source software wherever possible. Figure 5 shows the three highly-modified Jaguar UGVs from Dr. Robot[13]. Two of the UGVs are 4x4s and one has tank tracks (on the right). While each UGV came with an internal GPS receiver, the GPS was replaced with a better-performing NovAtel receiver and pinwheel antenna. The GPS antenna and IMU where moved outside of the main vehicle body. This reduced interference caused by the motors and the ground. An Asus netbook with a 1.6 GHz Atom processor running Linux (Ubuntu 12.04 LTS) was added to each vehicle. Webcams and Hokuyo URG-04LX LIDARs were also added to the UGVs.

Figure 5 also shows the three UAVs: 3DR Arducopter Hexa B's[14]. The Arducopters include onboard GPS/IMU, 2.4 GHz RC control links, 1.3 GHz video links, and 900 MHz communications links to the individual UAV's base stations. The UAV base stations were also operated on 1.6 GHz Asus netbooks running Linux. This effectively allowed the UAVs to have identical processing capabilities as the UGVs, but that processing was located offboard due to the hexacopter's limited payload capacity.

Additionally, a pan/tilt security camera was mounted to a tower and integrated into the system; several human actors wearing GPS backpacks were used as moving objects of interest; the user interface was run on a Windows 7 base station; and all wireless communications used standard 802.11g hardware.

The low-level autonomy was implemented in a fully distributed fashion onboard the Asus netbooks using the Robot Operating System (ROS)[15]. Custom Extended Kalman Filters were written for state estimation. Simple PID controllers were created to close turn-rate and velocity loops for waypoint tracking. These two simple algorithms were computationally efficient alternatives to ROS's standard slam_gmapping and move_base packages. Those packages were also incorporated to enable SLAM and advanced path following (including obstacle avoidance). When the more advanced ROS algorithms were run, the netbooks were taxed to their computational limits. However, the more insurmountable problem was that the Hokuyo lidars needed for SLAM did not function well outdoors during daylight.

The mid-level autonomy was also implemented in a distributed fashion on the Asus netbooks using ROS. The mid-level behaviors and metric estimators were custom-written, but also utilizing the ROS framework. The information they received and sent from the high-level autonomy used UDP datagrams broadcast at 1 Hz containing Google Protocol Buffers messages. This information format could be easily read and written by the high-level autonomy. The high-level autonomy's assignment algorithm is decentralizable, but due to time constraints was implemented in a centralized fashion using Matlab's Optimization toolbox, also running at 1 Hz.

Finally, the user interface was implemented using Sandia's UMBRA framework[16]. UMBRA provides a 3D model-based interface that was extended for the SAHUC implementation. An XBOX 360 controller enabled the human operator to teleoperate the UMSs whenever necessary.

4. PHYSICAL SECURITY DEMONSTRATION

SAHUC's potential for physical security applications was demonstrated at Sandia's Robotic Vehicle Range (RVR), figure 6. The RVR is a Sandia facility on Kirtland Air Force Base that enables Sandia to test robots for varying applications. It includes multiple buildings, towers, paved areas, gravel areas, dirt paths, and a fenced security perimeter; making it an ideal site to test how UMSs can aid in physical security.

All three UGVs were used in the demonstration, but only one UAV was flown. This was not due to technical limitations, but to ensure safety and comply with all pertinent regulations. The single flying UAV had a Sandia-trained aircraft operator. This operator acted as a safety pilot to take over manual RC control in the event of an emergency situation. The UAV also had a dedicated pilot-in-command (a FAA licensed private pilot) in one of the RVR's towers. The pilot-in-command monitored the UAV, communicated with the FAA control tower, and monitored the sky for other aircraft.

The pilot-in-command and aircraft operator fulfilled their required safety roles, but were not active participants in the system or demonstration. The human operator ran the Graphical User Interface from inside a designated control room within the RVR, having no line-of-sight to the UMSs being commanded. The three UGVs and three UAVs (only one of which actually flew) operated inside and just outside of the fenced security perimeter. Finally, a person acting as 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, figure 8.

The human operator began his shift by distributing the UGVs throughout the facility. A "patrol border" objective was created at the open gate because the operator knew this was a weak point in the current security situation. The tracked UGV began to immediately move back and forth along the open gate, streaming video back to the operator. Once that objective was assigned the human operator manually specified waypoints for the other two vehicles to spread them throughout the facility, figure 7. Then, a "visit point" and another "patrol border" objective were used to place the other UGVs at strategic locations (at the corner of a building and patrolling an entry point) in the RVR facility, figure 9.

The intruder attempted to sneak through the open gate past the UGV patrolling the Eastern gate. The wide-angle camera on the UGV detected the intruder as he tried to cross the patrolled perimeter. In an actual security application, machine vision processing would have been used to alert the human operator to the detection, and to continue tracking the intruder's location. However, for demonstration purposes this capability was simulated through a GPS backpack worn by the intruder, figure 8.

Upon detecting the intruder, the human operator created a "track target" objective. This was assigned to the UAV because of the UAV's ability to more rapidly reach and track the person, figure 8. The UAV continued to track the intruder until the intruder disappeared from sight by moving under a tarp, figure 9.

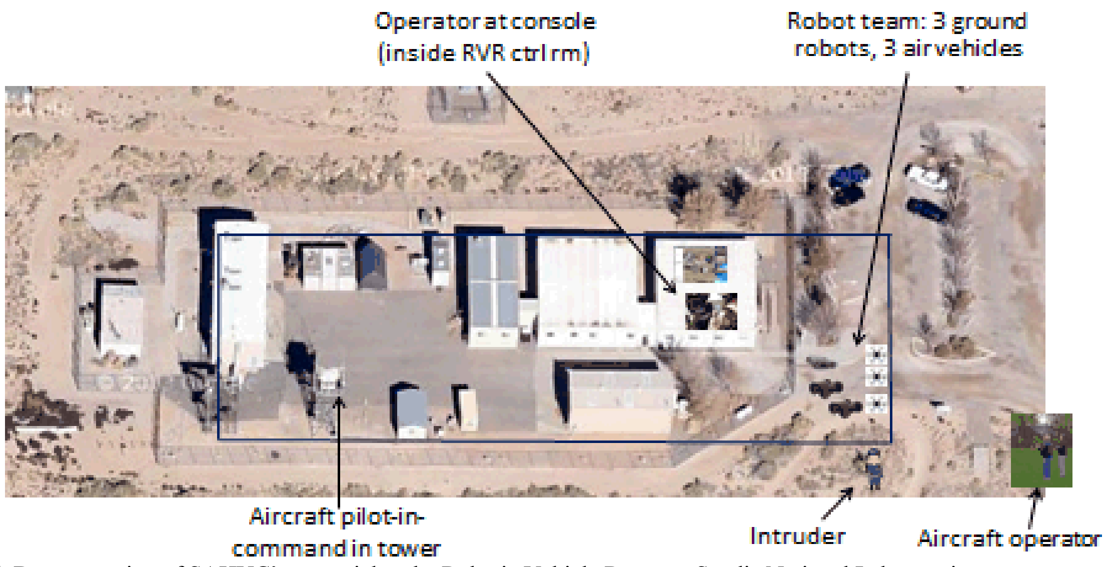

Operator at console
(inside RVR ctrl rm)
Robot team: 3 ground robots, 3 air vehicles
Aircraft pilot-in-command in tower
Intruder
Aircraft operator

Figure 6. Demonstration of SAHUC's potential at the Robotic Vehicle Range at Sandia National Laboratories.

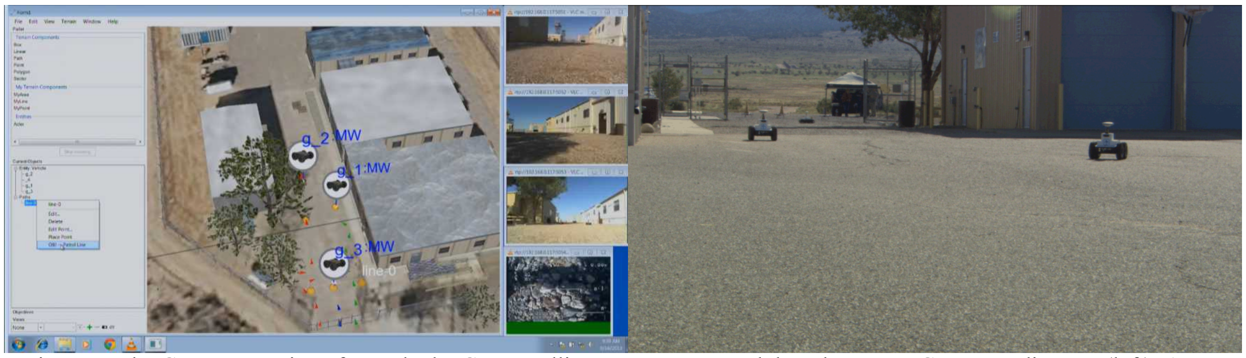


Figure 7. The GUI perspective of a tracked UGV patrolling an open gate and the other two UGVs spreading out (left). A photograph of all three UGVs underway (right).

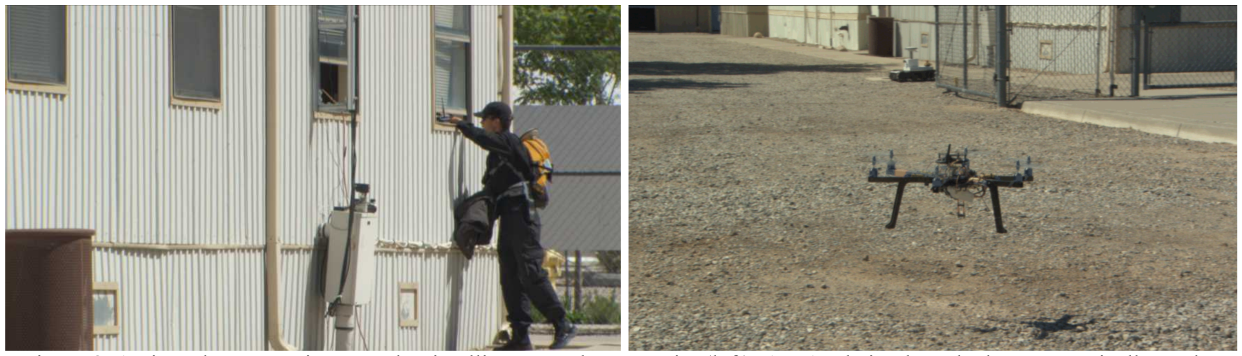


Figure 8. An intruder attempting to gather intelligence at the RVR site (left). A UAV being launched to automatically track the intruder, and the UGV performing the border patrol objective that detected the intruder in the background (right).

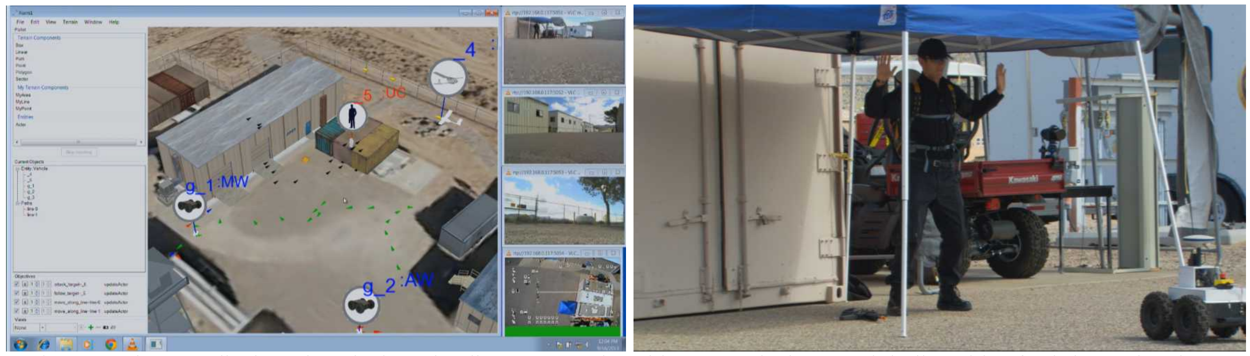


Figure 9. The GUI display when the intruder disappears under a blue tarp. The bottom-right live video feed shows that the intruder is no longer visible to the UAV (left). The intruder is then detected and confronted using a UGV (right).

The human operator created a new "look at point" objective to get streaming video of what was under the 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 UMSs could aid in the physical security of a high-consequence site. The need for heterogeneity was shown when the UAV could better track the intruder but could then not see under the blue tarp, whereas the UGV could easily see under the blue tarp but would have had trouble keeping pace with the intruder in the open. The impact of the autonomy was evident since the human operator was controlling 4 UMSs simultaneously, monitoring the live sensor feeds, while also explaining the system and answering questions to demonstration attendees.

5. CONCLUSIONS

The Sandia Architecture for Heterogeneous Unmanned System Control has demonstrated that small heterogeneous teams of UMS can be controlled by a single human operator to aid in securing high-consequence sites. Several autonomy algorithms have been adapted to emphasize timely responsiveness over optimality, producing a system that responds in seconds to changing security conditions. SAHUC's proof-of-concept implementation has illustrated that this is possible, even using comparatively inexpensive COTS hardware.

The flexibility within SAHUC to enable heterogeneous UMS to use heterogeneous algorithms is an important strength. As new algorithms become available and applicable they can replace the existing algorithms, promoting extensibility. As computation capabilities continue to improve, more advanced and computationally expensive algorithms may be run onboard the UMSs. These algorithms must continue to run quickly enough to maintain SAHUC's responsiveness. The distributed computation of the metrics required for the assignment algorithm is also a strength. The ability to plan paths and estimate metrics at varying levels of coarseness (depending on the UMS's computational capability) further aids in responsiveness. The major drawback to distributed algorithms is their need for high-quality communications. While this is true with SAHUC, the communications burden required to move SAHUC's information is significantly lower than streaming the live video feeds.

Future work will include larger scale demonstrations. These will potentially include several pan/tilt sensors acting as stationary UMSs. The inclusion of all three UAVs will further improve the performance of the system and yield more complete demonstrations. Currently the GUI displays the live video feeds next to the 3D map; overlaying the video directly onto the 3D representation will make it easier to intuitively see what each UMS is seeing and compare that to what it should be seeing (e.g. there should not be a black truck there). GPS tagging intruders with a backpack should be fully replaced with computer vision algorithms run onboard the UMSs. These algorithms may require the onboard computing to be increased to handle the additional workload. Finally, demonstrations should be run with intentionally degraded communications to evaluate how the overall system would perform under natural or man-made interference.

REFERENCES

[1] Intelligent Systems, Robotics, & Cybernetics, <http://www.sandia.gov/research/robotics/index.html> (21 January 2015).

[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] Scokaert, P.O.M.; Mayne, D.Q.; Rawlings, J.B., "Suboptimal model predictive control (feasibility implies stability)," *Automatic Control, IEEE Transactions on* , vol.44, no.3, pp.648-654, Mar 1999.

[11] Zeilinger, M.N.; Jones, C.N.; Morari, M., "Real-Time Suboptimal Model Predictive Control Using a Combination of Explicit MPC and Online Optimization," *Automatic Control, IEEE Transactions on* , vol.56, no.7, pp.1524-1534, July 2011.

[12] Linear Programming Relaxation, *Wikipedia*, <http://en.wikipedia.org/wiki/Linear_programming_relaxation> (17 December 2013).

[13] 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).

[14] 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).

[15] <http://jaguar.drrobot.com> (21 January 2015).

[16] < http://3drobotics.com/ > (21 January 2015).

[17] <http://www.ros.org> (21 January 2015).

[18] <http://umbra.sandia.gov> (21 January 2015).