Dynamic Task Sharing Within Human-UxS Teams: Computational Situation Awareness

Scott Grigsby(✉), Jacob Crossman, Ben Purman, Rich Frederiksen, and Dylan Schmorrow

Soar Technologies, Inc., Ann Arbor, MI, USA
scott.grigsby@soartech.com

Abstract. In current military operations, a team of human operators, often distributed across multiple locations, is required to manage even a single Unmanned Vehicle (UxV). For future multi-UxV control, effective dynamic task sharing strategies – the ability to quickly re-assign tasks and responsibilities between operators or between operators and autonomous systems - will vastly improve team coordination. However, for task hand-off to be executed effectively, the task assignee needs to be quickly brought up to speed with sufficient situation awareness to effectively handle their new tasking. We have implemented a system, called Computational Situation Awareness (CSA), that encodes the awareness maintenance process into the control station itself. CSA generates a computational "mental" model of an expert's SA and maintains multi-level awareness of the mission and state of the unmanned systems. This allows the system to predict information requirements and drive cueing and other mitigations for individuals and across the team. By tracking user tasks and system state related to those tasks, CSA builds an understanding of the task's progress thus enabling it to better determine what information the user needs to maintain task relevant SA without bogging them down. By encoding workload assessment and situation awareness into the operator control station itself, the control station becomes a partner with the operator (or team of operators) in making sense of the data. This enables it to manage task sharing and offloading and share information in terms that aids rather than distracts, thus improving each operator's mission effectiveness.

Keywords: Situation awareness · Workload · Human-Machine teaming · Decision support

1 Introduction

Supervisory control of multiple UxVs is a challenge, due to the cognitive burdens it places on the operator. These burdens include:

1. Staying aware of all important mission activities and events
2. Predicting outcomes and activities so as to proactively address them
3. Making decisions about how to best deploy aircraft
4. Shifting and regaining awareness when multiple activities occur simultaneously
5. Triaging and offloading tasks when overloaded
6. Recognizing faults or failures

Prior work suggests that, under the right conditions, supervisory control should provide an operator with the ability to control 4–8 UAVs simultaneously given the right conditions (i.e. that the level of automation is correct, and required situation awareness is maintained) [1]. Current systems fall far short of this level – most requiring multiple people to manage one aircraft.

The Multi Autonomous Ground-robotic International Challenge 2010 (MAGIC 2010) [2] provided an opportunity to test the limits of human supervisory control in a reasonably complex ground domain. In this competition, multiple robots executed ISR and dangerous object interaction tasks under the supervision of one or two operators. For this environment, Soar Technology, Inc. (SoarTech) developed the SAGE user interface designed to keep operators (and competition judges) aware of the overall mission as well as important events that were critical to the mission. Though only one of several factors that influenced the results, SAGE (along with a significant amount of robot autonomy [3]) helped a team of two operators efficiently control 14 ground robots. This results suggested an opportunity to apply new technologies to the problem of multi-vehicle control – specifically, technology that can enhance and expand operator situation awareness.

This prior work has led to the further research, development, and evaluation of what we call **Computational Situation Awareness (CSA)**. CSA is stand-alone software that maintains multi-level awareness of the mission and state of the unmanned systems. CSA Ingests (via networking protocols) data regarding the mission, UAV state, and some user-entered data. CSA then maintains an internal operating picture, detects task, failure, and mission-relevant events, and estimates user workload. The purpose of this technology is to help the operator continuously orient him/herself to the situation with the goal of making faster and better decisions.

2 Approach

Multi-UAV missions require a combination of global situation awareness (e.g., the state of the whole mission) and awareness of specific details such as the activity of a specific track, or the cause of a failure. Alternating between these types of macro and micro views of a mission is known to cause loss of situation awareness and thus can lead to poor decisions or change blindness.

To aid the user, CSA maintains situation awareness within the system itself and, unlike the human, CSA does not get distracted, attending equally effectively to all aspects of the mission. Unlike other systems, CSA maintains this awareness at multiple levels as defined by Endsley [3]:

1. **Perception:** Like most command and control systems, CSA maintains an internal operating picture (IOP) with the locations and states of each entity.
2. **Comprehension:** Unlike most other systems, CSA monitors this perception data for important relationships that matter in the mission. When discovered, these relationships trigger *events*. Events are a key output of CSA – they represent situations that require user attention. Events are maintained in a queue and are color coded to indicate priority. Events are CSA's means of orienting the user to critical situations.

Without CSA, the operator would need to use scan-and-assess techniques to maintain the same awareness making it difficult to maintain both macro- and micro-awareness.

3. **Projection:** CSA also projects some information into the future – for example, computing estimated task completion time. This projection data is integrated back into CSA's internal operating picture allowing events to be proactively triggered on predicted situations.

This multi-level IOP can be used for purposes beyond event detection. Our existing Lucid system uses this data to estimate the workload required of the user to execute the mission. This estimate associates a multi-dimensional workload model with each event and task in the IOP, and from the collection of these models, it computes an overall workload. This estimate is used within the system to automatically offload tasks to other users when overloaded.

CSA's ability to maintain its own internal operating picture can form the basis for situation awareness for both human users and autonomous entities. Within a decision support application, CSA would output its events using a smart interaction module to display alerts or events, allocate tasks, or otherwise inform missions, allowing the human user to take intelligent actions. Within an autonomous control system, CSA would be used to track and inform the entities decision processes directly.

2.1 CSA Architecture and System Integration

Computational Situation Awareness is implemented as a software module that ingests a broad range of state data and outputs filtered state data, events, and workload estimates. Figure 1 illustrates the CSA data flow.

The central component of CSA is the internal operating picture (IOP), which stores state data. This store is implemented as an in-memory store with separate indexed data structures for assets, tasks, tracks, and other relevant state items.

The IOP is maintained by a set of processes that are continuously processing incoming state messages from simulation (though the source is unimportant) and either (L1) simply storing them, (L2) making additional computational inferences on them, or (L3) making projections based on them. The Ln labels indicate the level of processing provided by these functions in Endsley's levels of SA terminology [4].

The basic SA processes set the stage for event detection. Event detection finds patterns in the IOP that match to mission, task, and failure situations that are of interest to the user. Some of these events are hard coded (e.g., the task start/completion tasks and the failure tasks). Others are informed by user requests (e.g., the force protection zone and restricted areas). These events are prioritized and output to the network where any component can subscribe to them. Each event is associated with a specific client (based on the task offloading work we will discuss in the next sub-section) and, thus, clients that do not "own" an event, display the event differently (grayed out, low on the event queue).

Events can be added to the CSA module relatively easily by simply implementing an event class and logic. There is support in the infrastructure for various spatio-temporal computations that can be reused to support new events. New event classes can easily be added within a day or two.

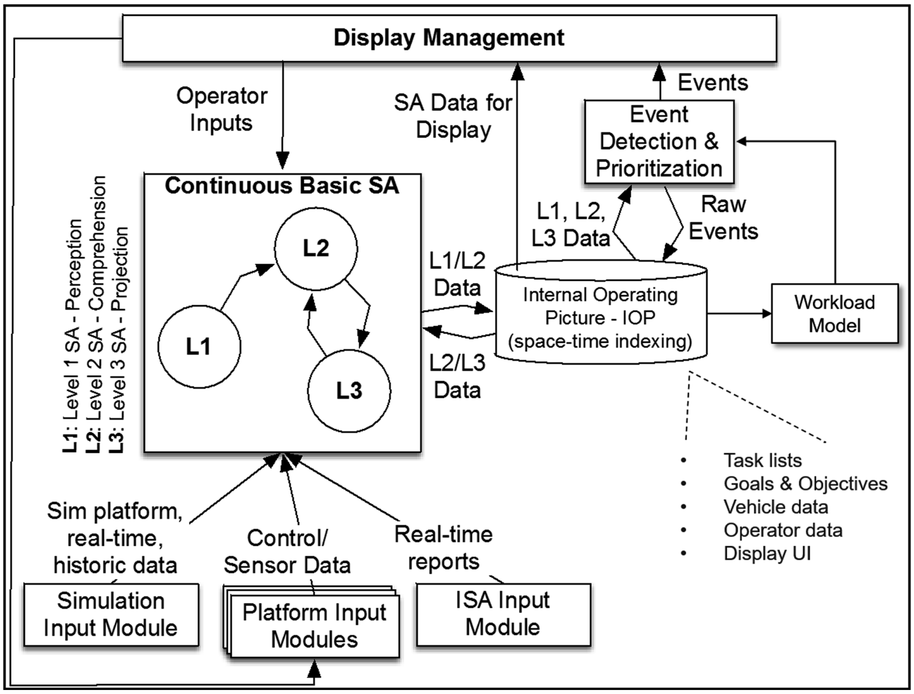


Fig. 1. CSA architecture

As an example, Table 1 summarizes the classes of events that were supported as of the end of our Lucid effort. The shaded rows highlight the events that the user can request explicitly. The "zone" based events are created in two cases: (1) when mission plan data is received force protection and restricted operating zones are created around objectives and dangerous areas, and (2) when the user creates a zone of interest using the polygon drawing tools in the UI. The track-based events are created when users select to "watch" a track. In these cases the user is alerted when new information about that track becomes available, when the track is lost, and when the track changes course significantly. These user-driven events are a critical component to CSA's utility as they let the user specify how the CSA system should help him/her. In our evaluation, these user-driven events were most valuable in helping the operators execute the mission.

2.2 Real-Time Workload Assessment and Offloading

The IOP is also used to infer the workload on the user in any given situation. When the workload is estimated to be above a single operator's capacity to execute effectively, events and tasks are offloaded to another operator. The basic workload assessment capability uses a 3-dimensional model to estimate workload in the cognitive, visual, and manual dimensions (these three dimensions are taken from Wickens [5], but here the auditory mode is not incorporated to simplify the model for initial implementation).

Table 1. Classes of events that CSA can detect (Lucid implementation)

Event name	Type	Description
Orbit point start	Task	Detect when orbit starts
Search route start/end	Task	Detects when route search starts and when end of route is reached
Search area start/end	Task	Detects when areas search starts and ends
Follow vessel	Task	Detects when tracking starts (failure event detects when it is lost
Asset task hasn't changed	Task	Detects when an asset has not been tasked recently (default: 5 min)
Force protection (user)	Mission	Detects when red/yellow tracks are discovered within a specified region
Restricted operating zone (user)	Mission	Detect blue forces in an unsafe area
Area of operation breach (user)	Mission	Detect that an asset has left an expected area of operation
New info available (user)	Mission	Detects when new data is available to track – Automatically created for any track being followed
Track change (user)	Mission	Detects when a track changes course (45 deg) or is lost – Automatically created for any track being followed
Fuel warning	Fault	Detects when a vehicle is at risk of not making back to carrier group with fuel levels
High/low altitude	Fault	UAV breaks an altitude boundary
Lost comms	Fault	UAV has lost communications
Near max comms range	Fault	Reaching limit of direct line-of-sight communication (Fire Scout-B)
Not making expected progress	Fault	Asset appears to be moving incorrectly (not following expected path)

The basic architecture for workload estimation is illustrated in Fig. 2. Events and state data (most importantly, tasks) are pulled from the IOP. For each event and task, the system looks up the workload model for that event and reads the expected

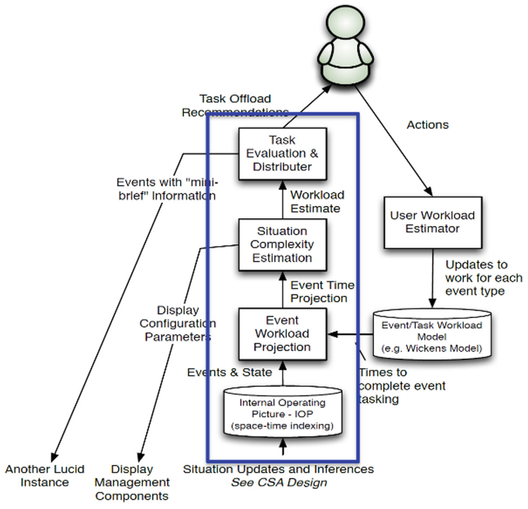

Task Offload Recommendations
Actions
Task Evaluation & Distributer
Events with "mini-brief" Information
Workload Estimate
User Workload Estimator
Situation Complexity Estimation
Updates to work for each event type
Event Time Projection
Display Configuration Parameters
Event Workload Projection
Event/Task Workload Model (e.g. Wickens Model)
Events & State
Times to complete event tasking
Internal Operating Picture - IOP (space-time indexing)
Another Lucid Instance
Display Management Components
Situation Updates and Inferences
See CSA Design

Fig. 2. Workload estimation architecture.

instantaneous workload value for that event at the given time. The expected instantaneous workload values are aggregated, using Eq. 1, to form an overall estimated workload. This value is fed into the task evaluation and distribution function where decisions are task distribution are made.

The equation used to compute the overall workload is as follows:

$$W(t) = \sum\nolimits_{i=1}^{N} max_{k=1,2,3}\{w_k(t - t_i)\} + g(t) \tag{1}$$

$$g(t) = 0.1 \times 1.25^N \tag{2}$$

Here W(t) is the estimated workload computed as the maximum of the accumulated component (e.g., cognitive, visual, manual) workloads. Component workloads are computed as functions of time, where each t_i is the start time for the ith event or task. The $g(t)$ component approximates the non-linear effects caused by multiple simultaneous active events (attention switching). Its constant values were tuned during laboratory tests such that workload values approximated to 1.0 when a user appeared to be overloaded. g(t) presents some problems as it scales exponentially over the whole range of N, which is not realistic and can overestimate the workload for high Ns. It works reasonably well for N < 15, but provides excessive estimates. A future version will likely replace the exponential with a sigmoid function to model a diminishing effect as N increases to large numbers.

This model is similar in concept to the Wickens' model [5], but differs in that it scales to an arbitrary number of tasks and events and accounts for change in workload over time such as temporal delays, workload ramp up, and workload ramp down. Our model also designed to be more precise in that it seeks to derive a number that can be compared to the user's full load level (e.g., 1.0 = fully loaded).

The decision when to offload is not straightforward. We found in our discussions with users during the evaluation that there are several ways tasks can be divided within a team.

1. They can be broken up purely based on their **timing** (e.g., round robin distribution, or secondary user gets all tasks after the main user becomes overloaded).
2. They can be broken up based on **load balancing** (e.g., the choice of which user gets which tasks is made so as to minimize user workload differences).
3. They can be made based on **geography** (e.g., each user takes care of a geographic segment)
4. They can be made based on **asset ownership** (e.g., each user gets events and tasks associated with the assets that he/she owns).

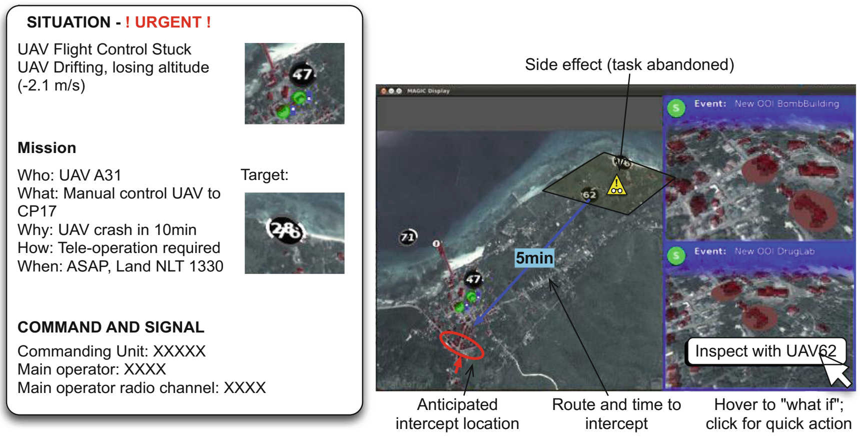
In practice, during our evaluation, operators used more than one of these approaches when they were given a choice.

We only had time and resources to select one offloading method for our initial implementation. We selected a variant of (2) for its simplicity. The variation was that the algorithm tries to load users one at a time – meaning it will not balance the load until at least one user is fully allocated. The idea behind this strategy is that it allows the secondary operator(s) to focus on other tasks while the primary operator is able to do the task alone. The algorithm is as follows:

```
1. Assign all unassigned events to the user with the highest
   workload (this is the primary user).
2. If this user's workload estimate is < 1.2 (combined) then
   stop, otherwise –
3. Find the maximum workload event assigned to the high workload
   user that has occurred in the last 30 s. (i.e. offload the event
   that is the biggest component of the workload.
4. Assign this event to the lower workload user if (event.work-
   load < 0.9 (high_user.workload - low_user.workload))
```

We note a couple of key points regarding this algorithm. First, events/tasks are not offloaded until the workload level reaches 1.2. This allows for the lack of precision in the workload model numbers. They are not intended to be correct to a single decimal place. Second, it moves tasks to other users from largest to smallest. The idea is to take the major focus tasks away from the primary user, so the primary user can continue to maintain global SA.

A crucial aspect of task offloading is maintaining situation awareness of the task parameters during handoff. Here CSA provides the appropriate mission parameters for the secondary operator. When the task is assigned or re-assigned, it is accompanied by a clickable "mini-brief" and recommendation with information drawn from the CSA IOP (Fig. 3) to enable the operator to quickly come up to speed on the task.



SITUATION - ! URGENT !

UAV Flight Control Stuck
UAV Drifting, losing altitude
(-2.1 m/s)

Mission

Who: UAV A31
What: Manual control UAV to CP17
Why: UAV crash in 10min
How: Tele-operation required
When: ASAP, Land NLT 1330

Target:

COMMAND AND SIGNAL

Commanding Unit: XXXXX
Main operator: XXXX
Main operator radio channel: XXXX

47
246
Side effect (task abandoned)
MAGIC Display
Event: New OOI BombBuilding
Event: New OOI DrugLab
5min
62
71
Inspect with UAV62
Anticipated intercept location
Route and time to intercept
Hover to "what if"; click for quick action

Fig. 3. Sample mini-brief (left) and recommendation panel (right) for task offloading

2.3 Architecture and Sample System Integration

The first implementation of CSA was within a system, called Lucid, to enhance operator situation awareness when executing control over multiple unmanned aerial vehicles (UAVs) to fulfill an ISR mission.

The Lucid system forms an innovative operator control unit (OCU) with a goal of decreasing operator: UAV ratios. Our specific research goals were:

1. Achieve operator:UAV ratios of 1:4 or better
2. Minimize operator interaction time
3. Maximize allowable neglect time (time the operator can neglect the system without degrading mission performance)
4. Maximize operator situation awareness
5. Lessen primary operator workload by automatically distributing tasking

Lucid was implemented in two pieces. First, CSA and workload estimation were combined into a single software component that was executed within a Java virtual machine. These technologies were then integrated with a supervisory control interface (RaptorX). Display and control capabilities were encapsulated within a plugin that resided in a RaptorX front end. These two components talk with each other using the Lightweight Communication Protocol (LCM). A proxy at the RaptorX end converts LCM traffic into native RaptorX TCP CommPath traffic so that the user interface plugins can send and receive data. The system as a whole is integrated with our own simulation environment, called SimJr, which provides simulated real-world data feeds and allows us to test and evaluate the system. The interface to SimJr is also a network interface using LCM.

3 Evaluation and Results

To test the efficacy of our CSA implementation, we implemented a rigorous evaluation process on the Lucid prototype. Our evaluation was designed to test several claims. Each claim is framed as a comparison between an operator using Lucid (the evaluation case) and a user using Raptor X without Lucid (the control case).

1. Using Lucid, operators will interact less with the system. Interaction here is defined to include control tasks and manipulation tasks.
2. Using Lucid, operators will be able to attend the mission less frequently in order to do other tasks.
3. Using Lucid, operators will have increased situation awareness of the mission, operating area, and asset state.
4. Using Lucid, operators will complete missions more effectively. We will measure effectiveness in terms of high value individuals identified, classification accuracy, reaction time, and task failures.

3.1 Evaluation Design

The Lucid evaluation was executed as a comparative study using a single control (or base) case and a single evaluation case. Both cases used the same scenario design and simulation and RaptorX as the core user interface. Communication between operators occurred only via a chat window. The independent variable for the study and the difference between the two cases is presence or absence of the Lucid capabilities, which are presented to the user as extensions of the Raptor X capability. Table 2 compares the features available in each configuration.

Operators were provided with a laptop computer with Raptor X and the Lucid plugin installed. Separate machines were used to run the simulation. Manual data was

Table 2. Evaluation configurations

	Control case	Evaluation case
Users	• Primary operator • Supporting operator	SAME
Human communication	• Text chat	SAME
Simulation	• SimJr running battlegroup/harbor scenario	SAME
Secondary task	• Elementary math using tablet	SAME
Front end	• Raptor X • Assets Commands • Polygon overlay drawing	• Raptor X • Assets Commands • Polygon overlay drawing • Event detection/queue • Situation mini-briefs • Decision overlay (if possible)
Task sharing	• Ad hoc using chat	• Auto event passing • Manual event passing (via chat)

collected (1) by the operators using predefined web forms/spreadsheets and (2) by the experiment director when required. At the end of each day, the team met in a hot wash to summarize what happened that day including collecting subjective feedback on performance.

Each test run (mission) has three main phases: planning/preparation (5 min), execution, and wrap up. The planning phase is short because the operators executed the same basic mission over and over (including training runs) and thus did not need much time to plan out activities. The mission execution phase was interrupted twice to measure situation awareness – once at 15 min and once at 30 min. During intermediate situation awareness testing the simulation was paused and the RaptorX screen blanked. For final situation awareness testing the simulation was paused, but the RaptorX screen remained visible. The wrap-up phase consists of structured questions and gathering of data from that mission. Each session typically ran 75–80 min.

Two operators executed each mission: a primary operator and a support operator:

Primary Operator: The primary operator is responsible for the overall mission including situation awareness, classification of tracks, and control of UAVs. The primary operator can offload tasks to the support operator when desired.

Support Operator: The support operator is responsible for any tasks the primary operator request for the secondary operator to execute.

The only exceptions to this structure were three runs made near the end of the evaluation when a second operator was not available. In these cases, a primary operator executed the mission alone.

Metrics

We collected a wide range of data associated with each of our experimental claims. A substantial volume of data was collected in the form of automatically generated logs from the RaptorX platform, including user actions with timing, and from the simulation (to capture ground truth of entity locations and timing of simulated events). For subjective workload measurement, we used an HTML version of the NASA Task Load IndeX (TLX) [6, 7], a widely employed multi-dimensional rating procedure that derives an overall workload score based on a weighted average of ratings on six subscales: Mental Demands, Physical Demands, Temporal Demands, Own Performance, Effort and Frustration. To assess situation awareness, we developed an adaptation of the Situation Awareness Global Assessment Technique (SAGAT) [4]. We supplemented the standard SAGAT procedure where subjects place entities on a map to demonstrate their awareness of the geographic situation with scenario-specific questions about mission-relevant conditions such as threats to own force and status of adversary activities.

User's Mission and Tasks

Overall, the operators were responsible for monitoring activity at a port. They were given initially 4, and as the mission progressed as many as 6 UAVs to execute the mission. These assets could be used to obtain tracks of vessels and ground vehicle in and around the port. Each track was associated with a set of property data (e.g., location, size, cargo, type), the completeness of which is dependent on the quality of the sensor reading on the vehicle. All operators were given the following mission brief:

> *"You are responsible for monitoring activity in a port city. You have some UAVs at your disposal and can use them to obtain tracks of surface vessels and ground vehicle in and around the port. Each track has some associated data, the completeness of which is dependent on the quality of the sensor reading on the vehicle. Intelligence reports suggest that adversary forces intend to move explosives and other equipment out of the city by sea. Past experience suggests that operations like this begin by moving explosives from a bomb-making factory to a training site, then moving to a dock where they are transferred onto a ship. Ships used for such transfers usually arrive only shortly before the contraband coming from the training site is to be loaded. The locations of the training site and two or three bomb factories are known, as are the locations of all the docks. It is not known which bomb factory will supply the training site, or which dock will be used for the transfer. An informant reports that the ship to be used is the Trojan Steed, which can be distinguished by its white color, cargo of large crates, and like new condition."*

Figure 4 provides an overview of the mission geography including the overall mission area view with the carrier group to the southeast.

Figure 5 is a close-up view of the land region where ground vehicles moved. These figures are taken from the RaptorX display.

Within the general activity of the harbor, the user is to detect three events:

1. A vehicle arriving at a known training site from one of two known bomb labs. The operator was required to identify the source location for this vehicle.
2. A vehicle leaving the training site at some point and moving to a dock to meet up with the Trojan Steed (sea vessel). The operator was required to identify whether or not the ground vehicle did meet up with the Trojan Steed.
3. Any vessel approaching the carrier group. The scenario supported carrier threats from either an opponent force destroyer or a fishing vessel. The operator had to correctly identify which vehicle threatened the battle group (as soon as possible).

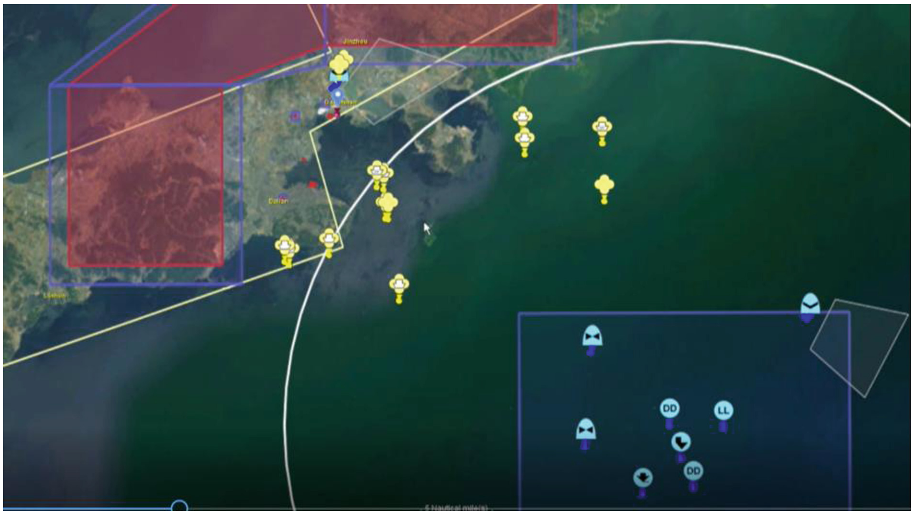


Fig. 4. Overall mission region

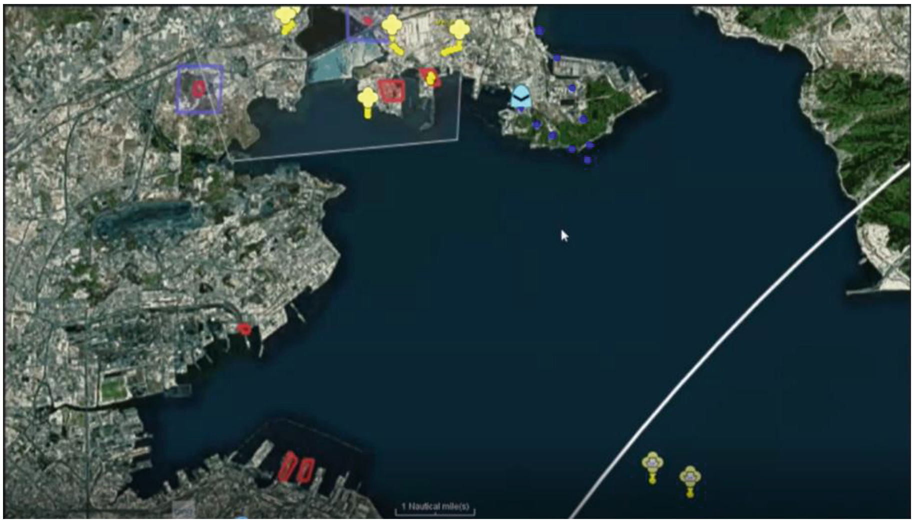


Fig. 5. Close-up of harbor area.

In support of detecting these mission-level events, the operator also attempted to complete these supporting actions:

1. Identifying the track of the vehicle that moved from a bomb site to the training area.
2. Identifying the track of the vehicle that moved from the training site to the dock or some other location.
3. Identifying the ship named “Trojan Steed” and its track.

We developed two different movement models – one for the sea and one for land. The sea vessels were spawned at both fixed and random locations in the sea (most at the beginning of the mission) and within the port. They each selected random destination and moved toward those destinations at rates dependent on the vessel type. The land vessels were spawned randomly along a road network and were given random destinations. Movement speeds were constant. The high value individuals (HVIs) were also randomized, but in a constrained manner:

- **Bomb Vehicle:** The vehicle moving from the bomb lab to the training camp. This vehicle spawned at a random time early in the mission. It source was randomly selected between two different bomb lab locations (one on the east, the other on the west).
- **Dock Vehicle:** The vehicle moving out of the training camp, possibly to a dock. This vehicle spawned a random (but constrained) number of seconds after the bomb vehicle arrived at the training camp. Its destination was randomized such that 50% of the time it went to the dock where the Trojan Steed was headed, while the other 50% of the time it went somewhere else.
- **Trojan Steed:** The sea vessel of interest begins out of port, but moves into the port to a dock during the mission. The Trojan Steed was randomly selected to be one of two sea vessels, one to the east and one to the west of the port mouth. Other clutter sea vessels with similar attributes were placed near each location. The Trojan Steed's dock destination was assigned randomly at startup (dependent in part on whether it would meet the dock vehicle that run).
- **Military Destroyers and Threatening Fishing Vessels:** Two destroyers patrolled the mouth of the port. At the beginning of the mission a threat was randomly selected among these vessels and a fishing fleet out of the port. The threat vehicle (either the military destroyer or a fishing vessel) would approach the carrier group over time.

To achieve the mission objectives the user can do two primary things: command UAVs and inspect/update information about tracks. The user can also execute supporting actions that help make decisions and commands easier. Table 3 lists the actions that a user can execute. Some of these activities can only happen when the user is using the Lucid plugin to RaptorX.

In addition to the main tasks, various situations can arise that require operator attention. Table 4 lists the situations that may occur during a mission and the expected operator response (or type of response). In many cases the operator has freedom in the details of the action taken.

Finally, the operators were asked to execute a secondary task whenever their primary task was not urgent or all consuming. The secondary task was the game "Tetris" executed on a tablet computer. The purpose of this secondary task is twofold:

1. To ensure that the user minimizes work on the main task (i.e., doesn't do "filler" tasks).
2. To estimate the allowable neglect time.

At the beginning of each scenario, the operator was asked to execute the secondary task as much as possible, running up as high a score as possible. At the start of the

Table 3. Available user actions in the Lucid and non-Lucid configurations.

Action	Description	Lucid only
UAV commands (speed and altitude can be set for each)		
Go to point	Go to a point in space and orbit or hover. Active sensor remains forward looking	
Observe point	Go to point or circular route where it is possible to keep sensor coverage on the point	
Search area	Fly in a circular pattern through an area such that every point in the area will be observed by the sensor at least once	
Search route	Fly along a route with the sensors oriented forward along the route	
Follow surface vessel	Position asset to keep vessel in view of its sensors. Move in order to keep vessel in view	
Track annotation		
View track information	The user can select a track and view the known information for that track (e.g., location, type, etc.). Some fields may be unknown	
Mark track ID	The user can set the ID to one of the known HVIs	
Mark track force	For unknown entities, the user can mark the force of the track (i.e., Hostile, Suspected, or Unknown). Tracks marked as Hostile and Suspected have different colors so that they appear visually clear	
Mark track intent	For the foreign military surface vessel, the user can select the track and set the intent either "Patrol Harbor" or "Approach Battlegroup"	
Map markup		
Create waypoints	The user can drop a waypoint on the map and assign a type to it: "Control" for use in tasking UAVs and "Location of Interest" to mark the main mission tasks	
Create routes and areas	The user can define routes as a series of waypoints and give them names. These can be used for route-based tasks	
Other		
Chat	The user can chat in a standard chat window with the other operator	
Event management		
Create watchbox	The user can create a box or radius and configure it to be watched by the system	X
Create watch on track	The user can mark a track to be watched for certain behaviors	X
Move event and asset control to other station	The user can select and offload an event (with an asset, if desired) to the other operator	X

Table 4. Failure events that could occur during a mission.

Situation	Symptom	Timing	Expected response
UAV runs low on fuel	Observe the fuel level in the asset details display	Fuel loss is permanent	Return the vehicle to the carrier group. If desired, launch reserve aircraft
UAV loses communication	Vehicle will appear to "freeze" for a period of time. The vehicle will continue its last action	Random. Comms loss may occur for 30 s to 2 min	Monitor for comms reacquisition. Possibly task another entity to complete that vehicle's mission
UAV navigation failure	Vehicle will drift away from planned route s and altitudes	Random. Navigation failures will repeat occurring periodically for up to 30 s each time	Possibly task another entity to complete that vehicle's mission
UAV breaches area of operations	Vehicle will be present in an area designated as "no fly" in the mission. Vehicle is removed from mission ("shot down")	This is not a random event, it only occurs when the operator loses track of the vehicle	Launch another vehicle or continue mission with one less vehicle
Track lost while attempting to auto-track	Aircraft will stop following sea vessel and just orbit the last known location	Permanent until re-acquired	User must direct asset to location where it can see the vessel to track and tell it to begin tracking again

mission the operators press the "start" button on the secondary task application, and pause when not executing this task. When the mission is complete, the operators end the secondary task and report their score and time. In practice, some operators did not correctly report their score/time so some data was lost.

3.2 Evaluation Results

The overall finding of our evaluation is that Lucid had a moderate, positive effect on primary **operator efficiency and effectiveness**. Further findings include:

1. Mission Effectiveness
   (a) With Lucid, users had a higher effectiveness on the ground tracking task. Ground tracking was by far the most difficult aspect of the mission.
   (b) With Lucid, users more effectively kept HVIs in sensor view.
2. Interaction
   (a) With Lucid, the primary operator issued fewer commands
   (b) With Lucid, the primary operator executed fewer track views
   (c) With Lucid, the primary operator sent less chat to the secondary operator
3. Situation Awareness
   (a) With Lucid, subjective situation awareness was higher
   (b) With Lucid, there were significantly more correct SAGAT answers

4. Neglect Time
   (a) With Lucid, the primary user spent more time on the secondary task (result not significant)
   (b) One user was able to execute secondary tasks while operating alone

Here we highlight four of the more interesting findings from the evaluation relevant to CSA, workload monitoring, and task offloading.

Primary operators using Lucid selected and viewed tracks data much less frequently (Fig. 6 left). This effect is statistically significant with a 35% reduction in track inspection ($p < 0.05$ for the 3-tailed T-test). We hypothesize that this is mainly due to Lucid's CSA providing some of the scanning/check tasks that the user would have to do in the baseline system. Because of Lucid's alerts for tasking, failures, and especially mission issues (like track changes), the operator could focus on one area for a period of time without missing as much.

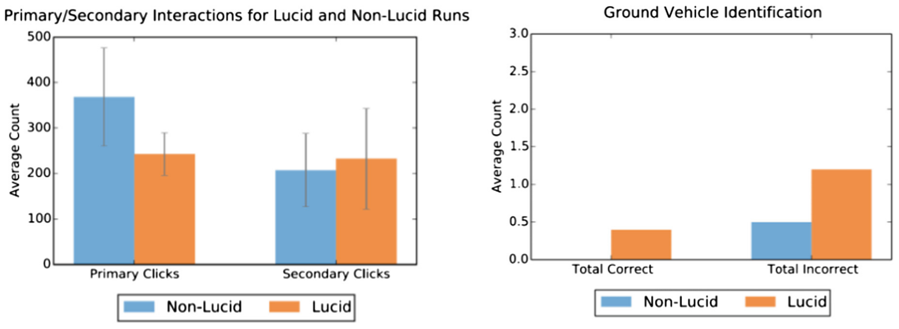


Fig. 6. Task counts for track-views (left) and ground vehicle identification (right).

Another interesting finding was that, overall, users attempted to identify ground tracks much more frequently when using Lucid than without (Fig. 6 right). In each mission, there are two ground vehicles to identify, and using Lucid the operator almost always made an attempt to identify both. Without Lucid, the operator only attempted to identify a single ground vehicle every 2 runs. Our conclusion is that Lucid aided the user in executing the ground-tracking mission, mainly through its alerts on tracks near the areas of interest, but also through its general reduction in required interaction time, providing more time to analyze tracks. The error rate was high using Lucid - on average, the user misclassified roughly one ground vehicle on every Lucid run, while correctly classifying roughly one ground vehicle every 2 runs. However, classification of ground vehicles was very difficult given the dozens of clutter vehicles and the many possible variations on the sources and destinations of the HVI vehicles. Thus, this error rate is not a surprise.

A surprising result is that the operators (especially the primary operator) issue far fewer chat lines when using Lucid than without (Fig. 7 left). For the primary user, the reduction is 44% with $p < 0.05$ using the usual 2-tailed T-test. We did not hypothesize that Lucid would reduce chat. We are uncertain as to why we see this effect. We believe that there are at least three possibilities: (1) a combination of the events and offloading reduced the need to communicate basic information about what was happening and where it was occurring, (2) the primary operator took more control during the Lucid runs and had the secondary operator do less, and (3) the primary operator, due to the stream of events felt more overwhelmed or engrossed in the details that he/she chatted less frequently. We cannot make any final conclusions as to which of these is true given our data, but we believe alternative (1) is most likely by elimination. Alternative (3) is not supported by the subjective evaluations (especially the TLX scores which show no difference in subjective workload between Lucid and non-Lucid runs). For alternative (2), we broke out the tasking activities by primary and secondary operator (Fig. 7 right). If alternative (2) were true we'd expect that for Lucid runs the secondary operator would issue many fewer UAV tasks than in the non-Lucid runs. This is not the case, in fact, the opposite is true (though not significantly so). This leaves us with alternative (1) as being most likely, and it is supported by the subjective and objective increase in SA shown above. This suggests that Lucid may be helping with shared situation awareness as well as individual awareness.

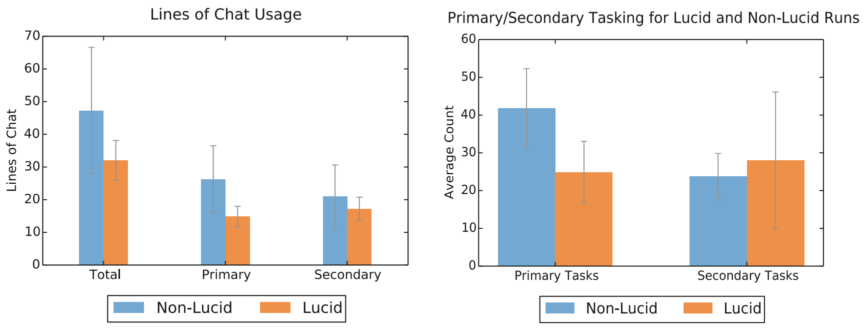


Fig. 7. Lines of chat (left) and workload estimate distributions (right)

Finally, as previously stated, Lucid did not have an effect on subjective workload based on the TLX measure – users felt workload with or without Lucid was comparable. This is not surprising in and of itself as the goal of Lucid was not to reduce workload per se, but to use workload measures to help balance task loading and allow the primary user to do more and be more effective.

4 Conclusions

The Lucid effort made significant strides toward developing a computational situation awareness capability that can aid a user in maintaining mission awareness and that can reduce some of the tedious work associated with executing ISR missions.

Our most important finding is that Lucid's CSA functionality appears to significantly reduce the manual interaction and intra-team interaction required for ISR missions using multiple UAVs. We hypothesize this effect is caused by (1) a reduction in the scan/check process required to maintain situation awareness and (2) better targeted tasking (few tasks issued, better sensor coverage). However, users did not report lower workload. This suggests that users were trading one type of work for another. In this case, we believe that Lucid reduced the user's manual workload allowing the primary user to focus more attention on cognitive and visual aspects of the problem. Our hypotheses is supported by the fact that operators using Lucid were able to execute the most challenging parts of the mission, ground target tracking, more often and more effectively when using Lucid.

This insight leads us to conclude that CSA's event detection and queuing capability, can be an effective complement to many standard common operating picture displays. CSA adds a layer of situation awareness and appears to improve the user's orientation to important mission items. It also enables a user to manage the mission better by reducing interaction (especially for SA tasks) and increases user focus on the critical tasks.

5 Future Directions

To achieve greater performance improvements (e.g., 2–3x), it would be most beneficial to combine CSA's capabilities with other improvements such as adding visualizations to the user interface that are specifically designed to take advantage of both event detection and workload estimation. Furthermore, while CSA's event detection appears to be a useful capability as implemented, further work to expand and refine this capability would allow it to be applied to more situations and have a larger impact. Potential areas for improvement include:

- Making events role specific and adding offloading strategies based on role, geography, and assets.
- Enhancement of the user-specified mission events. The users most liked these events and it is likely that these events were the biggest factor in the CSA's positive effects.
- Expand events to include compound events (combinations of situations that warrant attention).
- Further integrate event detection with UI elements. We spent relatively little time configuring the user interface to highlight event elements. Other related work suggest that such visualization improvements could increase the impact of the event detection system.

Finally, while we were able to implement a rudimentary workload estimation capability, Lucid's workload model would benefit from additional refinement. In particular, improvements to the workload model could be accomplished by using the evaluation data as a basis for model tuning/learning. For example, each workload function could be modeled as a probability distribution and the system could learn the parameters of these distributions using user activity. This would be most effective if combined with physiological sensors such as eye trackers, heart rate monitors, and EEG/MRI to input real-time objective of workload in addition to the task estimates.. Additionally, the following activities could improve the utility of the model

- Integration of user activity into the online model (e.g., the clicks)
- Implementation of UI configuration changes based on workload
- Use of workload to decide when to use autonomy rather than human control

Acknowledgements. This research was sponsored by a Small Business Innovation Research (SBIR) award through NAVAIR PMA (contract #N68335-14-C-0015). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied of the U.S. Navy or U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

1. Cummings, M.L., Bruni, S., Mercier, S., Mitchell, P.J.: Automation architecture for single operator, multiple UAV command and control. Int. C2 J. **1**(2), 1–24 (2007)
2. Crossman, J., Marinier, R., Olson, E.: A hands-off, multi-robot display for communication situation awareness to operators. In: International Conference on Collaboration Technologies and Systems, Denver, pp. 109–116 (2012)
3. Ranganathan, P., Morton, R., Richardson, P., Strom, J., Goeddel, R., Bulic, M., Olson, E.: Coordinating a team of robots for urban reconnaissance. In: Proceedings of the Land Warfare Conference, LWC 2010, Brisbane (2010)
4. Endsley, M.R.: Toward a theory of situation awareness in dynamic systems. Hum. Factors **37** (1), 32–64 (1995)
5. Wickens, C.D.: Multiple resources and mental workload. Hum. Factors **50**(3), 449–455 (2008)
6. Hart, S.G., Staveland, L.E.: Development of NASA-TLX (task load index): results of empirical and theoretical research. In: Hancock, P.A., Meshkati, N. (eds.) Human Mental Workload. North Holland Press, Amsterdam (1988)
7. Hart, S.G.: NASA-task load index (NASA-TLX): 20 years later. In: Proceedings of the Human Factors and Ergonomics Society 50th Annual Meeting, pp. 904–908. HFES, Santa Monica (2006)