

# CARACaS multi-agent maritime autonomy for unmanned surface vehicles in the Swarm II harbor patrol demonstration

Michael T. Wolf, Amir Rahmani, Jean-Pierre de la Croix, Gail Woodward,  
Joshua Vander Hook, David Brown, Steve Schaffer, Christopher Lim, Philip Bailey,  
Scott Tepsuporn, Marc Pomerantz, Viet Nguyen, Cristina Sorice, and Michael Sandoval

Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA

## ABSTRACT

This paper describes new autonomy technology that enabled a team of unmanned surface vehicles (USVs) to execute cooperative behaviors in the USV Swarm II harbor patrol demonstration and provides a description of autonomy performance in the event. The new developments extend the NASA Jet Propulsion Laboratory’s CARACaS (Control Architecture for Robotic Agent Command and Sensing) autonomy architecture, which provides foundational software infrastructure, core executive functions, and several default robotic technology modules. In Swarm II, CARACaS demonstrated higher levels of autonomy and more complex cooperation than previous on-water exercises, using full-sized vehicles and real-world sensing and communication. The core autonomous behaviors to support the harbor patrol scenario included Patrol, Track, Inspect, and Trail, providing the capability of finding all vessels entering the patrol area, keeping track of them, inspecting them to infer intent, and trailing suspect vessels. Significantly, CARACaS assumed responsibility for not only executing tasks safely and efficiently but also recognizing what tasks needed to be accomplished, given the current state of the world. Since the heterogeneous USV teams shared world model that evolved, such as due to (dis)appearance of vessels in the area or a change in health or availability of a USV, CARACaS replanned to generate and reallocate the new task list. Thus, human intervention was never required in the loop to task USVs during mission execution, though a supervisory role was supported in the autonomy system for mission monitoring and exception handling. Finally, CARACaS also ensured the USVs avoided hazards and obeyed the applicable rules of the road, using its local motion planning modules.

**Keywords:** maritime robotics, multi-agent autonomy, unmanned surface vehicles, autonomy architecture

## 1. INTRODUCTION

With the application of unmanned surface vehicles (USVs) likely to remain a key component of the naval defense technology roadmap,<sup>1</sup> the development of a multi-purpose autonomy architecture has taken an increasingly important role. Defense organizations foresee applying USVs in a variety of missions, such as intelligence, reconnaissance, and surveillance (ISR), submarine track-and-trail, high-value asset protection, mine countermeasures, and patrol. Ideally, USVs will be able to share the vast majority of their software code bases—in particular, the *autonomy architecture*—enabling new and modified missions via configuration rather than writing entire new software modules for each mission type. Further, the autonomy architecture should inherently support the teaming of multiple USV agents, as many of the envisioned missions entail cooperative autonomous USVs accomplishing missions jointly.

To this end, the Jet Propulsion Laboratory (JPL) has developed version 2 of its Control Architecture for Robotic Agent Command and Sensing (CARACaS), with an emphasis on industry standards and intrinsic multi-agent operations. This paper provides a high-level overview of the updated CARACaS autonomy architecture and specifically details its use in a full-scale 2016 demonstration known as “USV Swarm II”, sponsored by the Office of Naval Research (ONR). The Swarm II demonstration consisted of a four-USV harbor patrol scenario, in which the agents were responsible for patrolling an area for intruding vessels; tracking vessels in the area; inspecting

---

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the Office of Naval Research through an agreement with the National Aeronautics and Space Administration.

vessels, using sensors existing on only one of the USVs; and trailing selected vessels. While the demonstration highlighted a harbor patrol scenario, the CARACaS architecture developments themselves emphasized multi-mission infrastructure, using the harbor patrol mission as a challenging exemplar and motivating use case.

CARACaS, an autonomy software architecture with its origins in the NASA Mars Exploration Rover (MER) program, was first adapted for maritime robotics in 2004 and has been demonstrated on numerous USVs and unmanned underwater vehicles (UUVs) for ONR, the Defense Advanced Research Projects Agency (DARPA), and the Office of the Secretary of Defense (OSD). The architecture and its algorithmic components have been demonstrated in both single and multiple USV scenarios previously using the original architecture.<sup>2-4</sup> Previous work using CARACaS includes early sense-and-avoid technologies,<sup>5</sup> hazard avoidance algorithms that comply with the International Regulations for Preventing Collisions at Sea (“COLREGS”),<sup>6</sup> perception and planning algorithms for riverine patrol,<sup>7</sup> and the “USV Swarm I” high-value asset escort demonstration in 2014.<sup>8</sup> Swarm II was unique in that its mission complexity required a fundamentally different and more complete approach to multi-agent cooperation and provided a test case for the corresponding updated CARACaS architecture.

The remainder of this paper is organized as follows: Section 2 provides an overview of the CARACaS architecture and its approach to multi-vehicle cooperation. Section 3 describes the Swarm II mission and details the cooperative behaviors implemented in CARACaS to achieve it. Section 4 presents the results from Swarm II, and concluding remarks on solution efficacy and next steps are summarized in Section 5.

## 2. CARACaS ARCHITECTURE OVERVIEW

### 2.1 CARACaS Components

This section provides an overview of the various components that comprise CARACaS, with an emphasis on those playing a significant role in the Swarm II autonomy solution. Figure 1 depicts the functional components of a CARACaS deployment, grouped into notional sets of related roles. Vertical layering in this figure implies dependency and specialization, with upper layers building on top of lower layers and then encoding a particular application or deployment. Roughly, components on the left side pertain to the algorithmic decomposition and implementation, whereas components in the middle and right provide the connections between modules, support libraries for engineers and operators, and other ancillary functions. The remainder of this section discusses the contents of this figure from the bottom up.

Items grouped in MIDDLEWARE provide common libraries for modules to receive and share data from and to other sources. Most importantly for this area, CARACaS has adopted the Data Distribution Service (DDS) standard for its inter-process communication (IPC), an increasingly widespread standard for defense and robotics. DDS provides functions for both publish–subscribe and fetch operations in a common interface, with options on transport mechanisms and other quality of service settings. CARACaS makes use of both synchronous and asynchronous messaging options, and the use of *persistent* data on the wire, with keyed messages and built-in filtering, is central to the CARACaS world model implementation.

Capabilities grouped in ARCHITECTURAL SERVICES support all CARACaS modules with auxiliary functions. For example, CARACaS includes a Health Monitor that can be configured to send alerts to operators, abort autonomous operations, and/or notify other autonomous agents when there is a given fault. Also included are parameter management and data logging tools for consistency among modules.

CARACaS defines a default layout of what modules exist and what their roles are, as captured in the ARCHITECTURE DEFINITION, though there is ample room for extension and customization at all levels. A central element is the definition of the World Model, which is the union of everything the autonomous agent “knows”, both about the external environment (hazards, maps, moving objects, etc.) and internal and team states (agent pose, health, assigned tasks, etc.).

The AUTONOMY MODULES house the core planning and control algorithms and oversee the mission flow. These are discussed in Section 2.2, with specialization of particular mission models, commands, and behaviors categorized as AUTONOMY APPLICATION and discussed in Section 3.

The OPERATOR TOOLKIT and ANALYSIS TOOLKIT contain software that aid in testing, operations, introspection, and analysis of the system. These includes tools for simulation, software startup, runtime visualization, mission modeling, and the like.

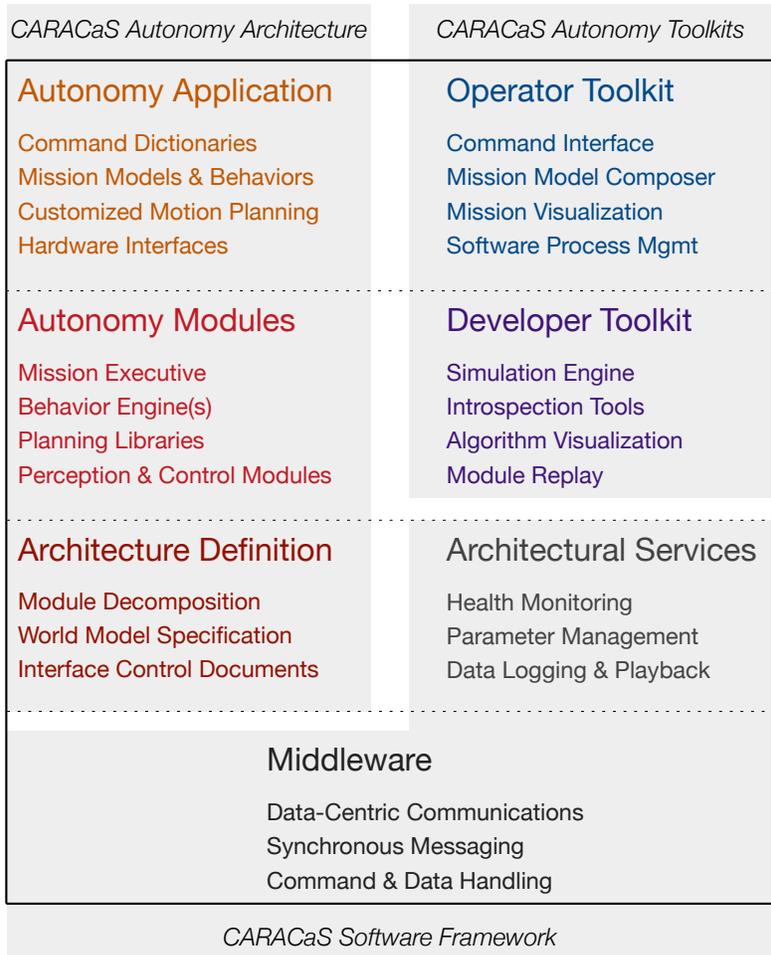


Figure 1. CARACaS functional components, divided into dependency layers. The top layer is the set of items typically specialized for a particular robot and/or application.

The particular behaviors and mission-level compositions of those behaviors are considered part of the AUTONOMY APPLICATION. These items tend to be customized on a per-domain basis, along with any customization of the default interfaces and perception modules that are likely to be specific to a particular platform, or at least particular domain.

### 2.2 Approach to Multi-Agent Task Recognition, Allocation, and Execution

All multi-agent operations in CARACaS are fully distributed—there does not exist a special vehicle to perform centralized processing. Within this distributed model, several approaches are possible to coordinate task allocation (i.e., which agents are assigned to different mission tasks). For the maritime domain, CARACaS coordinates vehicles via shared world modeling and implicit coordination.

Figure 2 diagrams the primary software modules and functional data flow for CARACaS. The autonomy planning and control elements reside in the Mission Executive and Behavior Engine. The Mission Executive is responsible for the planning pipeline from receiving the mission model from the operator and sending out what behavior(s) should be run for a particular. This includes determining what tasks are needed to complete the mission given the current world model state (*task recognition*); determining which agent should handle which task at what time (*task allocation*), across all agents and current and future tasks; and managing the behavior execution of the ego-agent’s timeline. Thus, all agents plan for both themselves and other agents, but they only execute the plan they make for themselves. This approach is appropriate for the amount of data needing to be

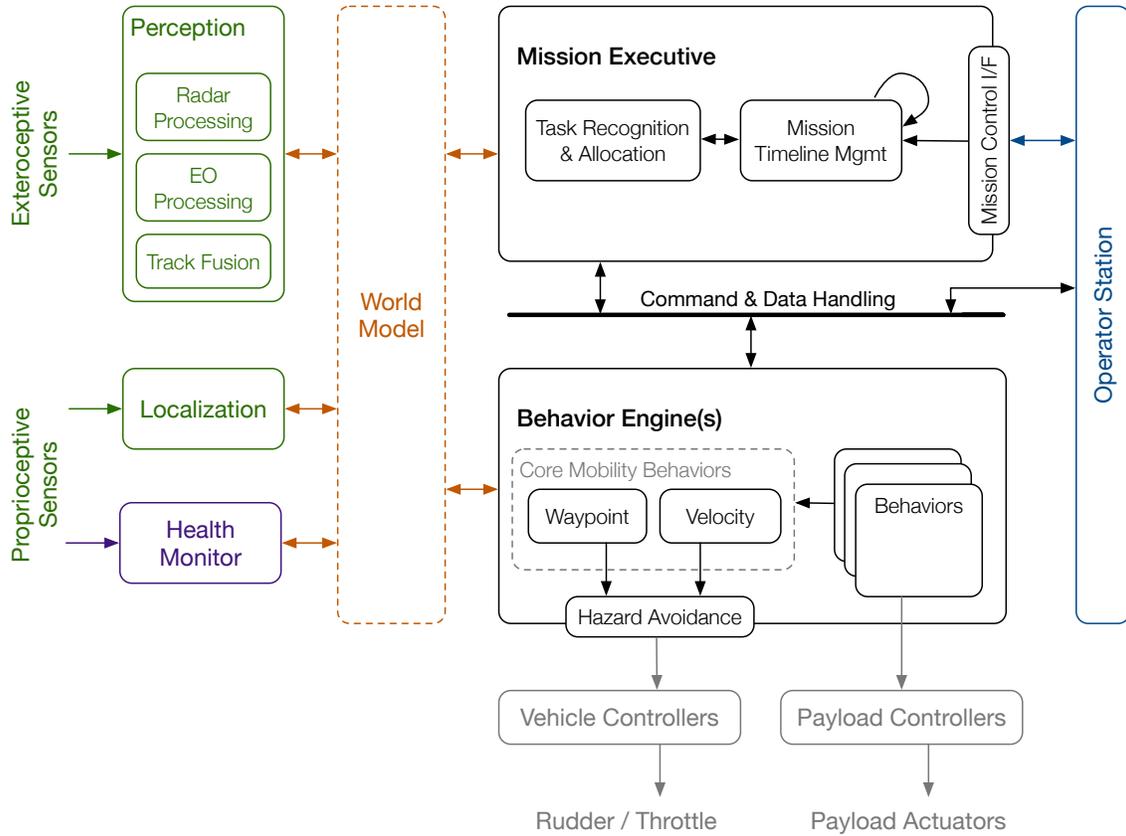


Figure 2. CARACaS functional architecture and information flow.

shared in the maritime domain and provides graceful degradation on communications issues, since in this model no explicit communication is required for the team to agree on task allocations.

The behaviors are responsible for *how* a task is accomplished (*task execution*). Behaviors are invoked in the Behavior Engine via the CARACaS Command and Data Handling (C&DH) interface. Top-level behaviors are invoked by the Mission Executive following the task allocation step. However, behaviors are typically hierarchical in nature—for example, an Inspect behavior may first call an Intercept behavior, which may call a Waypoint behavior—and thus the C&DH interface supports arbitrary calling of nested behaviors. The Behavior Engine provides the infrastructure for handling incoming commands, collecting needed data from the World Model, etc. The Behavior Engine also ensures that all mobility commands pass through a reactive hazard avoidance layer to keep the vehicle safe.<sup>6</sup>

For Swarm II, CARACaS was configured to share data between agents in two ways to establish a roughly synchronized world model. The first method was to fuse tracks in the perception subsystem. As different sensor processing components (radar and electro-optical [EO] cameras) produced candidate tracks, these tracks were shared among the USVs, along with navigation data; this accounted for the bulk of the radio traffic between agents. Tracks detected locally were correlated with the received data to generate a common track list among all agents. Software to accomplish this was provided by Daniel H. Wagner Associates.<sup>9</sup> The second method was to synchronize data directly within the world model, which provided a way for USVs to share data related to their own health and status and other locally observed updates. This capability was supported by Spatial Integrated Systems, Inc.<sup>10</sup>

### 2.3 CARACaS Autonomy Design Principles

There are several key motivating principles and benefits supported by the above design.

- Autonomy must be capable of event-driven operations in highly dynamic environments. Many missions, including Swarm II harbor patrol, cannot be completed with an *a priori* sequence or goal, such as a simple waypoint queue or achieving a given multi-vehicle formation. What tasks need to be accomplished for these real-world missions must be determined by the autonomous agents. Thus, this *task recognition* step is critical to achieve the level of autonomy desired for mission-capable teams, and is an element typically missing in other multi-agent studies.
- Autonomy must be robust to system and subsystem failures. When starting CARACaS, agents discover what resources are available and broadcast this information to others. This information is then available in the shared World Model for the Mission Executive to make task allocation decisions. This approach not only provides operational flexibility for startup sequencing but also provides a natural mechanism for robustness to failures. When the CARACaS Health Monitor catches a fault, all agents will receive updates on the modified set of available resources—such as a camera system no longer being available or an entire agent leaving the swarm. Since the Mission Executive re-plans periodically, the remaining agents will automatically adapt as a matter of normal processing.
- The autonomy architecture should support multiple missions. There are two elements in CARACaS’ approach to supporting a variety of applications in a particular domain. First, complex missions use behaviors as “building blocks”, where mission models are constructed through behavior composition. This allows behavior re-use and often enables different missions to be performed through configuration rather than coding. Second, CARACaS is designed to easily add new behaviors. As a behavior development platform, CARACaS has modularized each behavior and provided a small set of clearly defined APIs to build a new behavior and have it registered in the command dictionary.

### 3. HARBOR PATROL BEHAVIORS

#### 3.1 Swarm II Harbor Patrol Scenario

The goal of the Swarm II exercise was to demonstrate the use of cooperative USV autonomy in a harbor patrol scenario. A team of USVs, each a fully capable vehicle with GPS and local sensing with limited range, is tasked with patrolling a given region, as depicted in Figure 3. Any other vessel that enters this region (often referred to as a *contact*) must be tracked, inspected, and potentially trailed as long as it remains in the patrol region. The inspection operation requires that a USV equipped with cameras and on-board vessel classification software takes images of the contact to estimate whether the intruding vessel is of a particular type. If so, the vessel is considered “suspect” and must be trailed. If not, the vessel is “neutral” and must be tracked\*. A human supervises the operation and may override the autonomous classification and behaviors, but human input is never required for the USVs to continue the mission. The goal is to ensure that all vessels are detected and tracked, and that all suspect vessels are trailed.

#### 3.2 Swarm II Cooperative Behaviors

Fitting with the above mission, the Swarm II autonomy team generated focused work around four behaviors to complete the mission

- PATROL—cooperatively monitor a defined area
- TRACK—keep a given vessel within radar range of a USV
- INSPECT—dispatch a USV with cameras to classify a vessel
- TRAIL—closely follow a suspect vessel

This subsection provides an overview of these behaviors, including the conditions for task recognition and task allocation for each. Note, however, that the goal of this work was not to build optimized behaviors for true operational conditions but rather to create satisfactory implementations to demonstrate the concept of cooperative behaviors by USV teams.

---

\*To *track* an object, the object must remain within sensor range. To *trail* an object means to follow it at a particular offset. *Track* is primarily a sensing term, *trail* is a motion term.

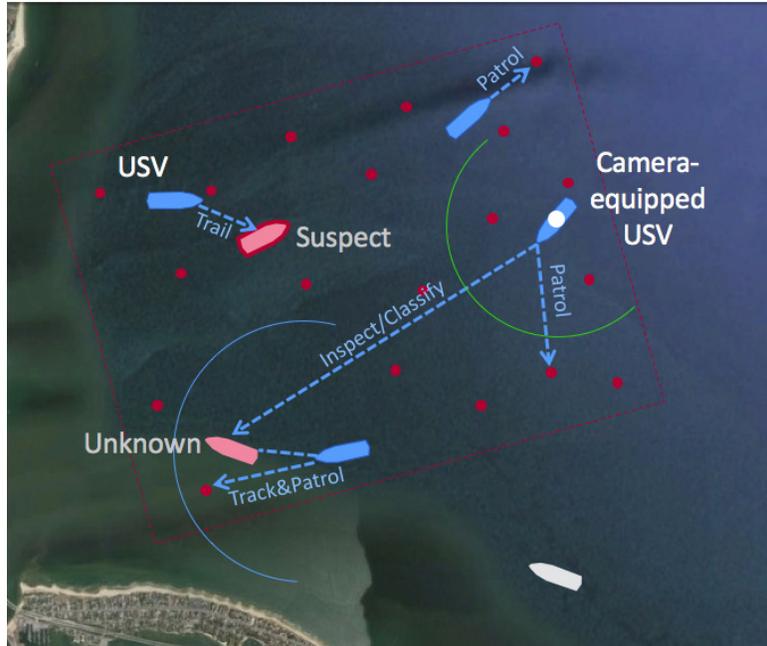


Figure 3. Diagram of the notional Swarm II harbor patrol mission scenario. Blue vessels represent USVs, one of which is equipped with cameras. Red vessels represent non-team members in the patrol zone. Red dots indicate notional patrol points the USVs aim to visit for area coverage. Blue and green circles signify that sensor ranges of the radar and cameras on the USVs are limited, but are not drawn to scale.

### 3.2.1 Patrol

When USVs are not subsumed by higher priority activities, they will patrol the region by visiting a set of defined locations, known as *patrol points*. The patrol points are defined *a priori* by the configurator of the mission based primarily on the geography of the desired patrol region and the USV sensor range. In Swarm II, the patrol points form a grid, but this is not a requirement. To *visit* a patrol point means to capture it within a USV's sensor range. Typically, the goal of the PATROL behavior is to minimize the maximum duration between consecutive visits, over all patrol points. The positions of all USVs, their availability to be assigned to PATROL, and all patrol point locations and visited times are considered jointly in an algorithm to assign points to individual USVs while maintaining a distribution and deliberative plan for subsequent patrol point assignments. Note that patrol points will be marked as visited even if it happens opportunistically during another behavior (such as TRAIL). As a USV visits a patrol point, it updates the World Model with the new visit time, which is then synchronized to the other USVs' World Models. The structure of this approach can be easily extended for randomized patrolling, weighting of certain subregions, and the like.

### 3.2.2 Track

All contacts in the patrol region must be tracked. The TRACK behavior keeps a given contact in the sensor field of view of the assigned USV. The closest USV that is not already engaged in TRACK is assigned to TRACK a new contact, with a preference not to assign a USV equipped with cameras, so it could be more easily freed to do INSPECT tasks. An interesting aspect of the TRACK task is that it may be accomplished simultaneously with other tasks. For example, a USV assigned to TRACK a neutral contact will also PATROL, with the TRACK behavior serving to constrain the USV to patrol points near the contact. To accomplish this, the CARACaS architecture was designed to reason over tasks that may be accomplished simultaneously, and which tasks require dedicated resource usage.

### 3.2.3 Inspect

When a new contact first appears, it is assigned a disposition of **neutral**. This new contact appearance generates not only a TRACK task but also an INSPECT task to attempt to determine if the vessel requires further attention

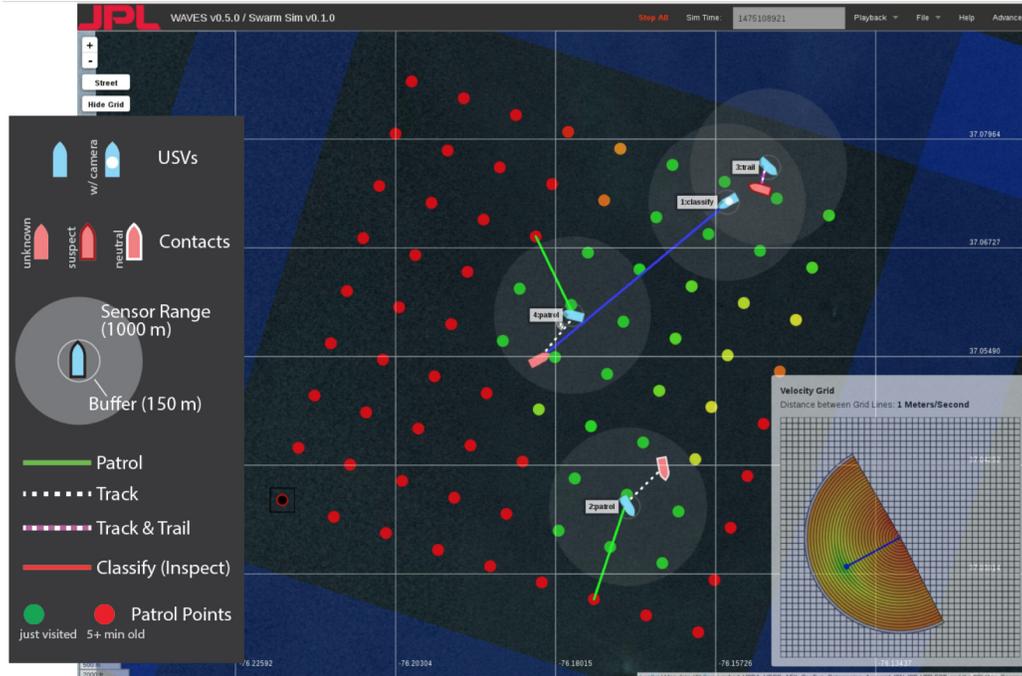


Figure 4. Operational area as seen via the CARACaS engineering display, with a legend inset on the left.

(i.e., assess its disposition). For the purpose of the Swarm II demonstration, visual inspection was used. (In real harbor patrol scenarios, the sensors and tactics used to inspect a vessel may be different.) The INSPECT task must be assigned to a USV with cameras and image processing software. Note that CARACaS, then, includes the capability of planning for heterogeneous teams—it assesses the resources required to complete a task and which agents have the resource available. The INSPECT behavior sends the camera-equipped USV to intercept the contact of interest and then holds the contact in the field of view of the USV’s forward-looking camera system (a JPL-designed “Hammerhead” stereo system). At this time, the behavior triggers JPL’s image processing software, known as the Contact Detection and Analysis System (CDAS), to locate the contact in images from the camera system and classify its vessel type. If it is of a predetermined vessel type, CDAS updated the World Model to indicate the contact is **suspect**; if not of the type, the contact was set as **neutral**. Additionally, a small sub-image of the contact and the estimated type was sent back to the operator. The disposition determined by CDAS (or possibly overridden by the operator) was synchronized to the World Models of the other USVs.

### 3.2.4 Trail

Contacts in the patrol zone marked as **suspect** must be trailed. The assignment logic for this was similar to TRACK—assign TRAIL to a nearby USV but preferably not a camera-equipped USV. The TRAIL behavior is a two-part maneuver. First the USV intercepts the contact, attempting to reach a (moving) waypoint set as a relative range and bearing from the contact. Once this location is acquired, TRAIL uses velocity-level commands to match the contact’s motion and maintain the desired standoff. Note that any USV in TRAIL will also be assigned to TRACK the contact, ensuring the TRACK requirement is fulfilled for the given contact.

## 4. RESULTS

The suitability of the CARACaS architecture for multi-agent applications, the ability of the Mission Executive to recognize and allocate tasks in a timely and distributed manner, and the performance of the harbor patrol behaviors were investigated in both simulations and on-water experiments. To this end, twelve scenarios with different degrees of difficulty—varying the number, disposition, speed, approach, and maneuvers of the intruding vessels—were designed and tested. Figure 4 shows the operational area as seen in the CARACaS engineering

display. As seen in the legend, the USVs, the allocated tasks, the contacts, the patrol points, and the patrol area boundary are visible in this display.

#### 4.1 Simulation Results (Idealized Perception)

Simulations were carried out on computers that replicate USV computer and software setups communicating over a local network with CARACaS’ multi-agent simulation environment. USV, sensor, communications, and environmental models were employed; however, perception remained idealized, without yet emulating false positives, dropped tracks, or significant discrepancies between the USVs’ local track lists (recall, Swarm II included software to maintain synchronized track lists). Under these idealized perception conditions, USVs consistently demonstrated correct and efficient behaviors for all twelve scenarios.

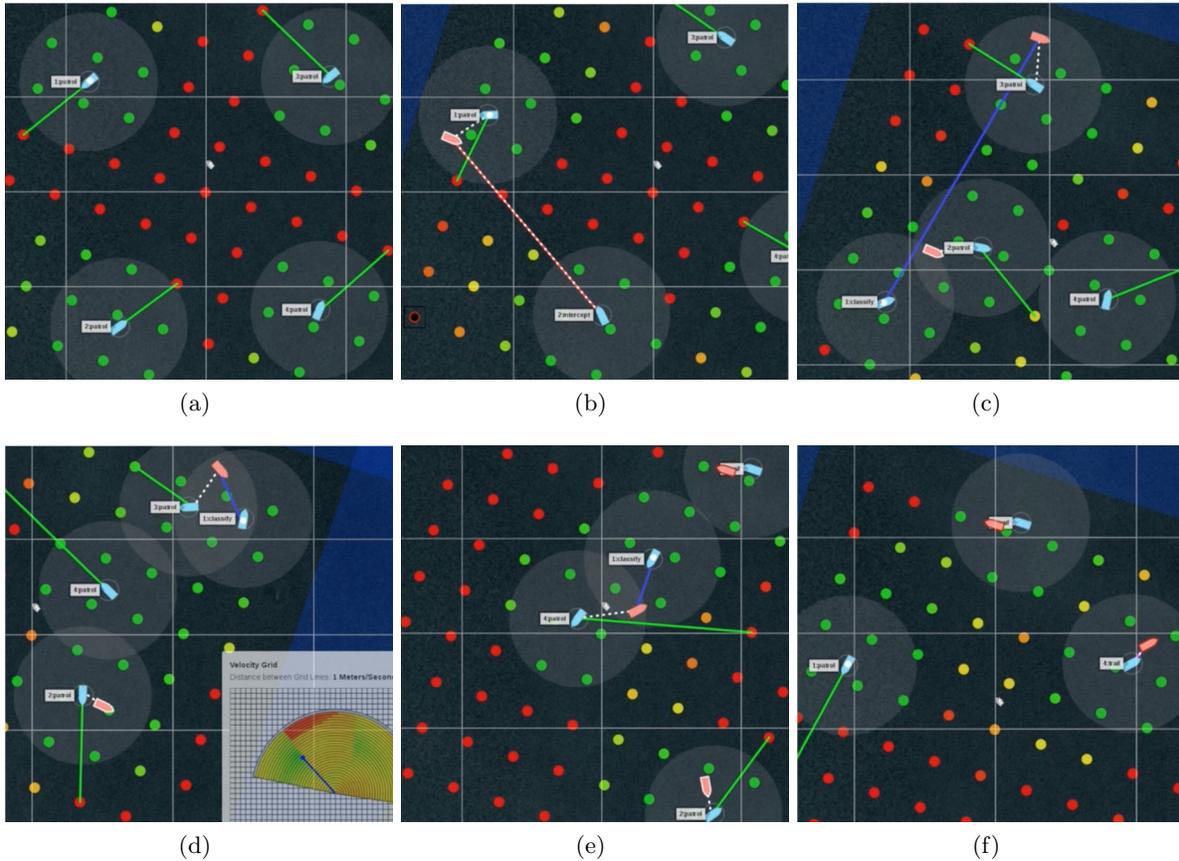


Figure 5. Swarm II behaviors during a simulation.

Figure 5 depicts one of the simulations of the harbor patrol scenario, a video of which is available with this paper. In this scenario, three contacts of interest pass through the patrol zone, and the USV swarm detects them and reacts to their presence. At the start of the simulation, all four USVs execute PATROL in the designated area (figure 5(a)). The patrol points appear as green when they have been recently visited, then become orange, then red as time passes without a visit. The first **unknown** contact of interest that enters the patrol area (figure 5(b)) is detected by a camera equipped USV and is immediately **INSPECTED** by the USV, resulting in a **neutral** disposition. Recall that **neutral** contacts must be **TRACKED**, which is done by the first USV; however, a second USV is assigned to **TRACK** the contact in the future to free up the camera-equipped USV for potential future **INSPECT** tasks, and so the second USV begins to intercept the contact. Figure 5(c) shows when a second **unknown** contact of interest is detected by the third USV at the Northeast corner of the patrol area. The USV **TRACKS** the contact while the camera equipped USV responds to intercept and **INSPECT** it (figure 5(d)). The velocity grid depicted on the bottom of this figure shows the hazard avoidance layer in action; the red cells are velocities



Figure 6. Swarm II USVs during the demonstrations at the Chesapeake bay.

that can lead to a collision with the contact. The INSPECT task results in this contact being labeled as **suspect**, and the third USV is assigned to TRAIL of the contact. A third contact crossing diagonally through the patrol area is detected (figure 5(e)) and INSPECTED to result in another **suspect** disposition. Figure 5(f) shows how two of the USVs are in TRAIL of the **suspect** contacts, while one USV is performing limited PATROL while in TRACK on a **neutral** contact and the camera equipped USV is in unconstrained PATROL.

#### 4.2 On-water Results (Noisy Perception)

On-water demonstrations were performed at the Chesapeake bay (figure 6) to test USV swarm’s effectiveness for harbor patrol in a mission realistic environment.<sup>11</sup> A mix of the twelve scenarios were executed over a month of on-water tests. Twenty runs were selected to be analyzed by a team of subject matter experts. Across these twenty runs, the four USVs operated over a total of 38 hours of autonomous operations.

Overall, the autonomy system performed well, with the USVs generally executing successful and safe operations. All assessed runs were deemed to provide effectiveness in the harbor patrol mission. The results indicate that most of the autonomy approach including task recognition, multi-agent task allocation, cooperative behavior execution, motion planning, and hazard avoidance was valid and effective, with robustness to communications issues.

However, the evaluation also highlights high dependency on the fused situational awareness picture and the adverse effects of perception noise on the overall system performance. The majority of the anomalies that impacted expected behaviors were perception related, such as track IDs changing and dropped tracks, likely stemming from radar performance issues. The observed behavior changes were generally of short duration (seconds) and were easily observable by the operator at the main Command and Control display. False positives caused the autonomy to react to inputs from perception that were not physically manifested (false, incorrect, etc.), resulting in less than optimal performance until these tracks were correctly dropped. Figure 8 illustrates how a false positive causes the USV to change its course during TRAIL to avoid a potentially unsafe situation.

The Mission Executive was successful in distributed task recognition and allocation. Despite perception and communication issues, USV plans were in agreement over 80 percent of the time and the team showed acceptable response time to new tasks. Behaviors also met or exceed their performance expectations. As an example, figure 7 depicts how a USV in trail closed on a **suspect** contact and stayed at the desired trail point behind the contact despite disturbances.

Some of the perception anomalies inadvertently showcase autonomy’s robustness to those class of anomalies and emergent behaviors. An example is depicted in figure 9, when the **suspect** contact being TRAILED by the green USV contact momentarily disappears. When the contact reappears, the green USV is already far behind. While the green USV tries to intercept the contact, the orange USV comes into a better position to intercept and TRAIL the contact, and hence a natural hand-off of the TRAIL takes place.

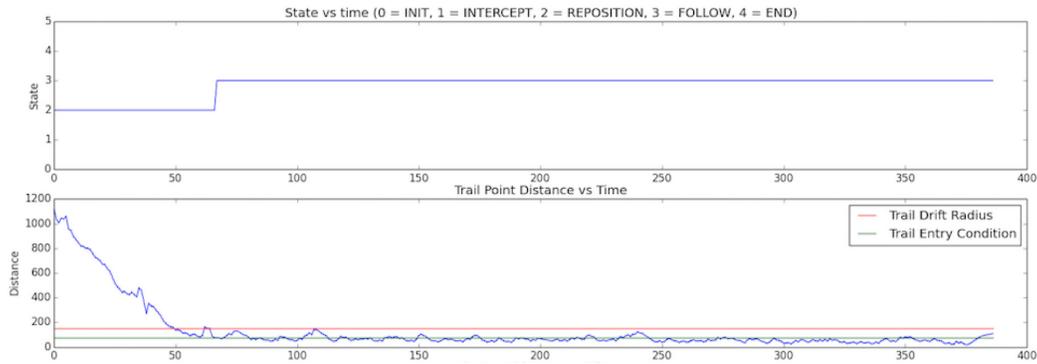


Figure 7. A USV in TRAIL of a contact of interest.



Figure 8. A false positive momentarily disrupts TRAIL.

## 5. CONCLUSION

This paper presented version 2 of autonomy architecture CARACaS and how it was employed to conduct a harbor patrol mission during the USV Swarm II demonstration. It showed how CARACaS supported multi-agent cooperation in a complex, event-driven mission without human input, in a way that was robust to intermittent communications issues as well as vehicle and subsystem faults. CARACaS assumed responsibility for not only executing tasks safely and efficiently but also recognizing what tasks needed to be accomplished, given the current state of the world.

In the Swarm II demonstration, the autonomy system performed well, with the USVs generally executing successful safe operations. The results indicate that the majority of the autonomy approach—including task recognition, multi-agent task allocation, cooperative behavior execution, and motion planning / hazard avoidance—was valid and effective, with reasonable robustness to communications issues. The primary difficulties resulted from sensor errors, especially since the mission scenario created a high sensitivity to false positives or mismatched tracks.

JPL plans to continue CARACaS updates to better support future USV Swarm autonomy. CARACaS

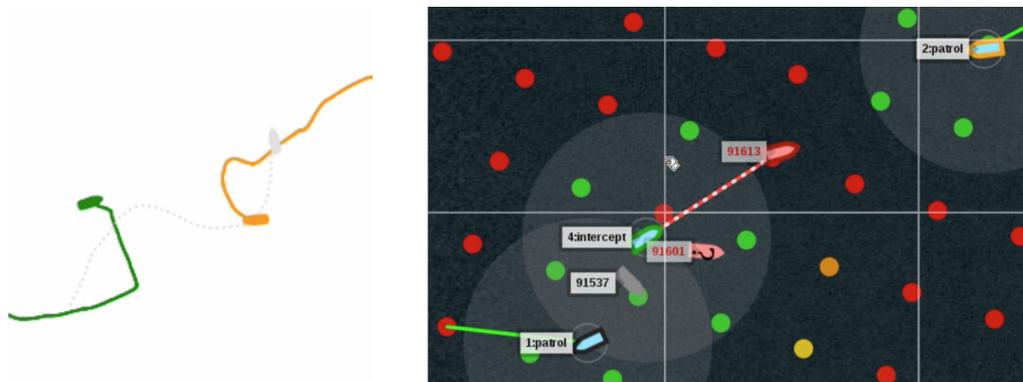


Figure 9. [Left] Ground-truth and [Right] moments before TRAIL hand-off between green and orange USVs.

situational awareness capabilities will be enhanced, adding robustness perception noise and delivering a broader set of situational awareness data products to use at different levels of planning and control, addressing the main difficulties during Swarm II. Also, the autonomy task allocation and execution elements will support more complex teaming scenarios, such as creation of subgroups and having closely coupled behaviors. Finally, further demonstrations will illustrate the “building block” approach of behavior composition to quickly adapt and tailor mission design.

## ACKNOWLEDGMENTS

The authors would like to thank Dr. Robert Brizzolara at the Office of Naval Research, Code 33, for his continued support. The authors also acknowledge Spatial Integrated Systems, Inc. (SIS), the Naval Surface Warfare Center (NSWC), Carderock Division, and D. H. Wagner Associates for their collaborations on this work. Field testing and demonstration were conducted by SIS and NSWC–Carderock.

## REFERENCES

- [1] Winnefeld, J. A. and Kendall, F., “Unmanned systems integrated roadmap, FY2011-2036,” Tech. Rep. 11-S-3613, U. S. Department of Defense (2011).
- [2] Huntsberger, T. and Woodward, G., “Intelligent autonomy for unmanned surface and underwater vehicles,” in [*OCEANS’11 MTS/IEEE KONA*], 1–10 (Sept. 2011).
- [3] Elkins, L., Huntsberger, T., Aghazarian, H., Monach, R., Crawford, S., and Fuller, J., “Sensors and autonomy development for the autonomous maritime navigation (AMN) system,” in [*AUVSI Unmanned Systems North America Conference 2008*], (June 2008).
- [4] Huntsberger, T., Trebi-Ollennu, P. P. A., Nayar, H. D., Aghazarian, H., Ganino, A., Garrett, M., Joshi, S. S., and Schenker, P. S., “CAMPOUT: A control architecture for tightly coupled coordination of multi-robot systems for planetary surface exploration,” *IEEE Trans. Systems, Man and Cybernetics, Part A: Systems and Humans, Special Issue on Collective Intelligence* **33**(5), 550–559 (2003).
- [5] Huntsberger, T., Aghazarian, H., Howard, A., and Trotz, D. C., “Stereo vision-based navigation for autonomous surface vessels,” *J. Field Robotics* **28**(1), 3–18 (2011).
- [6] Kuwata, Y., Wolf, M., Zarzhitsky, D., and Huntsberger, T., “Safe maritime autonomous navigation with COLREGS, using velocity obstacles,” *IEEE J. Oceanic Engineering* **PP**(99), 1–10 (2013).
- [7] Wolf, M. T., Assad, C., Kuwata, Y., Howard, A., Aghazarian, H., Zhu, D., Lu, T., Trebi-Ollennu, A., and Huntsberger, T., “360-degree visual detection and target tracking on an autonomous surface vehicle,” *J. Field Robotics* **27**(6) (2010).
- [8] Office of Naval Research, “Autonomous swarm.” <https://youtu.be/ITTVgk02Xw4> (Oct. 2014).
- [9] D. H. Wagner Associates. <http://www.wagner.com> (2017).
- [10] Spatial Integrated Systems, Inc. <http://www.sisinc.org> (2017).
- [11] Hsu, J., “U.S. navy’s drone boat swarm practices harbor defense.” IEEE Spectrum, <http://spectrum.ieee.org/automaton/robotics/military-robots/navy-drone-boat-swarm-practices-harbor-defense> (Dec. 2016).