AFRL-IF-RS-TR-2006-48
Final Technical Report
February 2006

MIXED-INITIATIVE CONTROL OF AUTONOMOUS UNMANNED UNITS UNDER UNCERTAINTY

Carnegie-Mellon University

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. M421

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2006-48 has been reviewed and is approved for publication.

APPROVED: /s/

CARL DEFRANCO
Project Engineer

FOR THE DIRECTOR: /s/

JAMES W. CUSACK
Chief, Information Systems Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	FEBRUARY 2006	Final Nov 2001 – Dec 2003

4. TITLE AND SUBTITLE
MIXED-INITIATIVE CONTROL OF AUTONOMOUS UNMANNED UNITS UNDER UNCERTAINTY

5. FUNDING NUMBERS
C - F30602-01-C-0219
PE - 62301E
PR - M421
TA - 00
WU - 01

6. AUTHOR(S)

Dr. Sebastian Thrun, Dr. Geoffrey Gordon, Mark Burstein, David Diller, Dieter Fox

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Carnegie-Mellon University
5000 Forbes Avenue
Pittsburgh Pennsylvania 15213-38980

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington Virginia 22203-1714

AFRL/IFSA
525 Brooks Road
Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

AFRL-IF-RS-TR-2006-48

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Carl DeFranco/IFSA/(315) 330-3096/ Carl.DeFranco@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***

The MICA program focused on changing the control and coordination of unmanned aerial vehicles from a need for two to four persons per vehicle to one person controlling five or more vehicles. This program developed techniques for hierarchical control using mixed-initiative planning guidance and control, taking a number of kinds of uncertainty into account at a fundamental level. These techniques focused on reasoning about uncertainty, including planning, belief tracking and communications with both human and automation. We developed this control model using Partially Observable Markov Decision Processes. The mixed-initiative interactions enabled users to describe constraints at multiple levels of the planning hierarchy. Techniques include visualization of the environment and optional speech input. The capabilities were demonstrated in a laboratory environment and on the program's Open Experimental Platform.

14. SUBJECT TERMS
Command and Control, Planning, Markov Decision Processes, mixed initiative.

15. NUMBER OF PAGES
39

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS

1. Introduction 1
2. Objectives and Overall Approach 3
3. Major Accomplishments 7
   3.1 Publications 7
   3.2 Year 1 9
   3.3 Year 2 10
4. Estimation of complex belief state 11
5. Planning under uncertainty 15
   5.1 Belief compression 15
   5.2 Market-based coordination 18
   5.3 What-if analysis for game-theoretic planning 20
6. Integration Architecture 22
   6.1 Planning constraint language 24
7. Mixed-initiative interaction techniques 26
   7.1 Visualization of planning policies 26
8. Reactive Agent Behaviors 27
   8.1 OEP Parameters 28
   8.2 Long SAM First 29
   8.3 Shoot Long SAM sites twice 30
   8.4 Face Threats 30
   8.5 Dive when Threatened 31
   8.6 Loiter Center 31
   8.7 Planner Parameter experiments 32
9. Robotic testbed for SEAD missions 33

LIST OF FIGURES

Figure 1. A gradient view of a policy. 6
Figure 2. The query-response algorithm in action. 13
Figure 3. The intent tracker. 14
Figure 4. A sample play from the playbook: four vehicles travel in formation until reaching a waypoint near the target, then spread out to approach the target from multiple angles. 15
Figure 5. A schematic representation of belief compression. 17
Figure 6. A snapshot of the execution process for our belief compression planner. 18
Figure 7. Combination of market-based coordination for multi-agent planning with what-if analysis for adversarial planning, as applied to our robotic testbed for the SEAD task. 20
Figure 8. Architectural framework for MICA U^4 interaction with testbeds. 23
Figure 9: Relationships among agents and situational views. 24
Figure 10: Elements of Constraint Language 25
Figure 11. Example constraint language expression 26
Figure 12. Year One views of single UAV (best) plan and overall policy. 26
Figure 13: Iconography of Plan/Situation Visualizer from Year 2 27
Figure 14. An example of a reactive behavior 28
Figure 15. Two of the robots used in our SEAD testbed. 34

1. INTRODUCTION

The DARPA Mixed-Initiative Control of Automa-Teams (MICA) Program has focused on changing the control and coordination of unmanned aerial vehicles (UAVs) from a situation where five or more people are required to manage one vehicle into a situation where tens of vehicles can be controlled by a single operator. This challenging problem requires both advances in distributed planning and control, and in approaches to mixed-initiative interaction between operators and advanced planning and control systems.

Autonomous and semi-autonomous vehicles will operate in highly dynamic, inherently uncertain and adversarial environments. Their commanders must plan their actions with only partial knowledge of their future environment, and the threats that will surround them. Vehicle plans must enable reactions based both on what they observe, and on what they *expect to observe* as they move, which itself may change over time.

CMU, with subcontractors BBN and the University of Washington, set out to develop a hierarchical control system that uses mixed-initiative planning guidance to control heterogeneous teams of vehicles. The scientific framework they laid out takes a number of kinds of uncertainty into account at a fundamental level.

The role of the CMU portion of the team was primarily the development of new techniques for reasoning about uncertainty, including methods for belief tracking, planning under uncertainty, and communication with other human and automatic reasoners at various levels of the mixed-initiative hierarchy. It also included developing the specification of the contractors' mixed initiative hierarchy, including the roles of various agents within the hierarchy and message formats for communicating among them. As such, the CMU team developed numerous new reasoning techniques which enabled new functionality in the context of the MICA effort; these techniques are detailed below, but include methods such as Rao-Blackwellized particle filtering, market-based planning, and the Query-Response algorithm for distributed belief tracking.

The role of the University of Washington was to work on highly efficient belief tracking algorithms and otherwise support the CMU effort.

BBN's focus was primarily on the mixed-initiative interaction between warfighters and the tools developed by CMU for planning and execution under uncertainty. BBN's

goal was to identify key concepts for incorporation into an easy-to-use, high-level command interface by which multiple teams of robots could be controlled in a wide array of combat situations. BBN also played an instrumental role as integrator of the planning and surveillance algorithms developed by CMU into a demonstration prototype that enabled those algorithms to be tested with Boeing's Open Experimental Platform (OEP).

A key technical claim of this project is that probabilistic multi-robot control systems can be developed which accommodate both uncertainty and user advice at many levels. We developed this control model using a hierarchy of **partially observable Markov decision processes (POMDPs)**. POMDPs are a probabilistic method for control under uncertainty. The approach bases control on the *information state* (knowledge state), not the actual world state. POMDPs generate plans that are really global control *policies* for acting in any observed state. Our approach uses efficient techniques to dynamically and frequently regenerate policies as new information is learned, and adjust control strategies accordingly, *e.g.*, by regenerating team formations if some agents are damaged, or adapting task assignments and paths in the presence of new threats.

Our team's approach to scalable hierarchies of POMDP controllers was based partly on prior related work with hierarchical MDPs. We demonstrated that hierarchical POMDPs can be more efficient, and can be used within a planning cycle for a team of ten or more UAVs to support adaptive to user-based command and control. Our hierarchy spanned path planning, to multi-vehicle coordination, dynamic team composition, and operator command and control. **Our mixed initiative user interaction model** and interfaces enable users to describe constraints at multiple levels of the hierarchy in conjunction with visualization of the environment and plans/policies generated in order to help users interpret problems that arise during both planning and execution. Planning policies and uncertainty information can be visualized on demand, or shown automatically when critical decisions are required. Speech input can optionally be used to provide additional planning constraints or explicit direction. Operational constraints suggested by users are incorporated dynamically during execution-time replanning. The overall effect is be to make timely and reliable interactive planning possible in the face of uncertainty, adversarial effects, and vehicle failures.

2. OBJECTIVES AND OVERALL APPROACH

The CMU/BBN/UW team's overall objective was to develop a hierarchical control system that uses mixed-initiative planning guidance to control heterogeneous teams of vehicles. We approached this objective by organizing our work into a group of several efforts:

- To develop algorithms for planning and situation awareness in the face of multiple kinds of uncertainty.
- To develop an architectural framework for integrating the mixed-initiative user interaction layer to the other layers of the hierarchical planning system and to the simulation testbed (OEP).
- To develop a language for relaying the objectives and constraints of users to the planning system.
- To develop a mixed-initiative interface to the hierarchical POMDP-based planning and execution system that supported a variety of mechanisms for visualizing the ongoing state of execution and planning policies for accomplishing the user's given objectives

We pursued all of these objectives consistently throughout the effort. Initially we focused on research and experimentation, while as the project progressed we spent correspondingly more effort on implementation and testing. The following sections will review our progress and achievements in each of these areas.

The primary objective of our efforts was developing a mixed-initiative control paradigm for planning under uncertainty. Four different sources of uncertainty need to be considered for this kind of combat planning and execution environment:

1. **Randomness:** Randomness in the environment causes uncertainty. For example, robots might malfunction at random; the speed and direction of a vehicle are random (within bounds), etc. Randomness can usually be accommodated using robust controllers, although randomness that exceeds planned-for bounds can require plan adjustments at multiple levels in the hierarchy.
2. **Incomplete knowledge.** Lack of knowledge causes uncertainty. For example, a controlled vehicle might not know the location of enemy vehicles or threats. Information gathering is an important planning strategy that needs to be addressed as part of the overall planning approach in order to manage the incomplete knowledge problem.

3. **Temporal phasing:** Not only must vehicles participating in a military plan address the uncertainties in their operating environment due to their own incomplete world models, they must also operate so as to respect the time phasing of the overall operation. Plans with timing constraints typically call for multiple agents to achieve objectives at approximately the same time, or at particular times relative to each other. Thus, it may be just as problematic to arrive early as it is to arrive late. Furthermore, when operating against an adversary that is also moving, getting to an objective too late is just as bad as not getting there.
4. **Adversarial effects:** The enemy will move (within a strategy) in ways that maximally impede the mission goals. Usually, the enemy strategy is unknown and therefore causes additional uncertainty. This type of uncertainty is the most difficult one, since the opponent can be expected to act in the worst possible way relative to one's own mission. We did not address this kind of uncertainty in the first two years of the project.

As we began this effort, no existing approach in the literature addressed all four types of uncertainty. Control theory and planning has focused on the first two types of uncertainty, while planning and combinatorial optimization focus on the third. Game theory has focused on the fourth (but not as much on the first three). Our team's objective was to provide a scalable approach for dealing with **all four types of uncertainty**, by bringing the best of today's planning, control, and optimization approaches and integrating them into a single mathematical framework that could be controlled by human operators. The mixed initiative framework would take advantage of the human operators' special abilities: in particular, humans can often be more sensitive to and able to adjust for the second and fourth types of uncertainty (incomplete knowledge and adversarial effects).

CMU's approach to planning under uncertainty is based on harnessing techniques based on partially observable Markov decision processes (POMDPs), extending this family of algorithms to address the need for distributed hierarchical control given the number of semi-independent agents (UAVs) that need to be controlled. In order to apply POMDPs to this planning problem, we require methods for distributed planning under uncertainty (so that agents can reason about the parts of the plan which are relevant to them and about which they have knowledge) as well as distributed situation awareness (so that all agents can coordinate their actions in support of the chosen plan). The methods we developed are detailed below; but they include Rao-Blackwellization of particle filters for distributed situational awareness; reasoning about communication (the

query-response protocol); market-based techniques for distributed planning; and fast planning techniques for POMDPs.

In support of rapid and effective user decision making for our planning process, our operator-planner communication development efforts (headed by BBN) focused on:

- Development of an internal language for representing user objectives and constraints on the planning problem for concise communication to the planning subsystem. (In collaboration with CMU)
- Experimentation with a variety of input modalities and mechanisms by which users could interactively communicate new objectives, constraints and advice to the planning system in a natural, convenient and timely fashion in order to respond to a constantly changing operational picture. These input mechanisms translated into the internal input language of the planner.
- Experimental development of plan visualizations based on observations constituting situation awareness, together with the output of POMDP planning, namely, action *policies.*

Policies are potentially high dimensional mappings of all possible observations to actions to take given those observations. The policies suggest optimal actions to take in all circumstances based on a Markov assumption relationship between observations and those future execution states. Effective visualizations for mixed initiative oversight of these planning systems must combine policy information with presentation of the sequence of activities that are anticipated as the best plan given the currently observed and anticipated conditions, *and* relevant known information about that current state.

It is extremely difficult to convey in a single picture all of the actions recommended by even a simple policy. Figure 1 depicts one visualization method we experimented with briefly. Here, the surface represents the essentially the utility of each possible observed state (the x-y plane) with the policy suggesting the action to take at any <x,y> to avoid threat areas (low utility) and reach the goal state (in red) is to move ‘uphill’, maximizing gradient ascent. As temporal effects are introduced into the domain, the state space grows in dimensionality extremely rapidly, and this kind of visualization is rendered useless. It was, in any case, not easily explained to naïve users. We return to a discussion of visualization techniques in a later section.

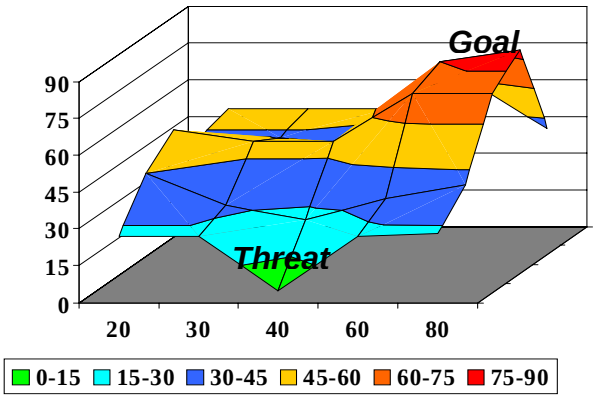

Figure 1. A gradient view of a policy.

3. MAJOR ACCOMPLISHMENTS

3.1 PUBLICATIONS

In support of the MICA effort we have prepared a number of publications in peer-reviewed journals, conferences, and workshops. Many of these publications are available from the CMU PI's website

`http://www.cs.cmu.edu/~ggordon`

while others are available on request.

Refereed journals and magazines

Nicholas Roy, Geoffrey Gordon, and Sebastian Thrun. Finding Approximate POMDP solutions Through Belief Compression. *Journal of Artificial Intelligence Research* **23**:1, 2005.

Vandi Verma, Geoff Gordon, Reid Simmons and Sebastian Thrun. Particle Filters for Fault Diagnosis. *Robotics and Automation Magazine*, special issue on Human Centered Robotics and Dependability, 2004.

Refereed conferences and workshops

Matthew Rosencrantz, Geoff Gordon, and Sebastian Thrun. Learning Low Dimensional Predictive Representations. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2004.

Rosemary Emery-Montemerlo, Geoff Gordon, Jeff Schneider, and Sebastian Thrun. Approximate Solutions For Partially Observable Stochastic Games with Common Payoffs. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS)*, 2004.

Joelle Pineau, Geoff Gordon, and Sebastian Thrun. Applying Metric-Trees to Belief-Point POMDPs. In Sebastian Thrun, Lawrence Saul, and Bernhard Scholkopf, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 16, 2003.

Curt Bererton, Geoff Gordon, Sebastian Thrun, and Pradeep Khosla. Auction Mechanism Design for Multi-Robot Coordination. In Sebastian Thrun, Lawrence Saul, and Bernhard Scholkopf, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 16, 2003.

Allison Bruce and Geoffrey Gordon. Better Motion Prediction for People-Tracking. In *Proceedings of ICRA*, 2004.

H. Brendan McMahan, Geoffrey J. Gordon, and Avrim Blum. Planning in the presence of cost functions controlled by an adversary. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2003.

Matt Rosencrantz, Geoff Gordon, and Sebastian Thrun. Decentralized sensor fusion with Bayes filters. In Christopher Meek and Uffe Kjaerulff, editors, *Uncertainty in Artificial Intelligence (UAI)*, volume 19, 2003.

Joelle Pineau, Geoff Gordon, and Sebastian Thrun. Policy-contingent abstraction for robust robot control. In Christopher Meek and Uffe Kjaerulff, editors, *Uncertainty in Artificial Intelligence (UAI)*, volume 19, 2003.

Nicholas Roy, Geoffrey Gordon, and Sebastian Thrun. Planning under Uncertainty for Reliable Health Care Robotics. In *Proceedings of the 4th International Conference on Field and Service Robotics (FSR)*, 2003.

Geoffrey J. Gordon. Generalized^2 linear^2 models. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 15. MIT Press, 2003.

Nicholas Roy and Geoffrey J. Gordon. Exponential family PCA for belief compression in POMDPs. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 15. MIT Press, 2003.

Joelle Pineau, Geoff Gordon, and Sebastian Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In Anthony G. Cohn and Georg Gottlob, editors, *Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

Mary Berna, Brad Lisien, Brennan Sellner, Geoff Gordon, Frank Pfenning, and Sebastian Thrun. A learning algorithm for localizing people based on wireless signal strength

that uses labeled and unlabeled data. In Anthony G. Cohn and Georg Gottlob, editors, *Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

Matt Rosencrantz, Geoff Gordon, and Sebastian Thrun. Locating moving entities in dynamic indoor environments with teams of mobile robots. In Tuomas Sandholm and Makoto Yokoo, editors, *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, volume 2, 2003. Won "best student paper" award.

Vandi Verma, Geoff Gordon, and Reid Simmons. Efficient monitoring for planetary rovers. In *Seventh International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, 2003.

M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In Proceedings of the AAAI National Conference on Artificial Intelligence, Edmonton, Canada, 2002. AAAI.

Carlos Guestrin and Geoffrey Gordon. Distributed planning in hierarchical factored MDPs. In A. Darwiche and N. Friedman, editors, *Uncertainty in Artificial Intelligence (UAI)*, volume 18, 2002.

Technical reports

Maxim Likhachev, Geoff Gordon, and Sebastian Thrun. ARA*: Formal Analysis. Technical Report CMU-CS-03-148, Computer Science Department, Carnegie Mellon University, July 2003.

S. Thrun. Simultaneous mapping and localization with sparse extended information filters: theory and initial results. Technical Report CMU-CS-02-112, Carnegie Mellon University, Computer Science Department, Pittsburgh, PA, 2002.

3.2 YEAR 1

Our efforts during year one were focused in several areas:

- We developed an XML-based language for communicating MICA domain planning constraints, operator objectives and advice to the Hierarchical POMDP planning system.

- We explored approaches to visualizing POMDP policies.
- We designed and implemented an architectural framework for user control of frequent interactive planning during execution in order to manage the uncertainty of a largely unknown adversarial situation.
- We designed new algorithms for situational awareness and belief tracking in the sorts of complex environments that we anticipated encountering in the OEP challenge problems.

By the end of the year, we had developed a demonstration platform with which we could begin testing both the capabilities of the POMDP planner and begin to examine and experiment with how users could control and advise such a planning system. We had also begun work on algorithms for the physical robotic testbed described below (the robots themselves were available from previous funding; our effort under MICA was limited to using them as a secondary evaluation platform for strategies in SEAD missions).

We also developed an initial POMDP-based planning algorithm and attempted to integrate it into our OEP tests; but we determined that this planner was going to be too slow to use repeatedly in a dynamic environment, and did not adequately support temporally ordered goals and waypoints. So, we began to design and integrate a new simpler planner for lower-level decisions at that time, and implemented autonomous agent behaviors to avoid threats discovered during flights, to give the system time to replan.

Finally, we started work on the technologies required for inter-agent communication; work on these technologies (such as market-based planning architectures, described below) continued throughout the project.

3.3 YEAR 2

During the second year of the project, our focus shifted as we had developed the basic integration infrastructure and began to experiment with the environment we had developed. Our goal was to determine which aspects of the planning and control problem were most appropriate for the user to do and which could be handled by the automated parts of the planning hierarchy.

We also, in response to government feedback, shifted some of the emphasis we had been placing on situational awareness to focus more on decision-making: since the OEP already contained simulations of the outcomes of belief tracking algorithms, there was less place than we had anticipated for some of the reasoning algorithms we had designed.

- We developed interactive mechanisms for users to specify pre-formed planning templates for such missions as SEAD missions, so that plans of multiple partially autonomous UAVs on teams could be coordinated.
- We developed tactical behaviors for UAVs within the action policies provided by the POMDP planner so that rapid reactions to unforeseen threats could be used to avoid weapon loss within the timeframe of the replanning.
- We developed the intrastructure for and performed experiments on the effectiveness of our overall framework.
- We explored a variety of techniques for operators to express planning constraints rapidly and effectively to support replanning, including preliminary demonstrations of an approach to speech input based on prior related work.
- We finished work on several algorithms which individual agents could use for planning under uncertainty.
- We refined and tuned our centralized planning algorithms for teams of agents.
- We continued the development of our market-based algorithms for decentralized planning.

Selected of these topics from both years are reviewed in the following sections.

4. ESTIMATION OF COMPLEX BELIEF STATE

Our research in belief state tracking treats two types of Rao-Blackwellized particle filters, conditional particle filters and FastSLAM. We have researched RBPFs in order to scale state tracking up to the high-dimensional, nonlinear models needed in the MICA domain.

This technology is relevant to some extent at all levels of the hierarchy, but it is most helpful at higher levels where partial observability is most prevalent. For example, the low-level reactive planner for a given unit can observe its own position via GPS, and

is told waypoints and areas to avoid by the team coordination planner. So, for practical purposes it has complete observability and can avoid solving complicated belief state tracking problems (which is good because it needs an extremely fast cycle time to keep up with the real world).

Conditional particle filters take advantage of the fact that, in our belief state (the posterior probability distribution over states of the world at the current time), there are often many statistical conditional independence relationships. For example, if we knew the exact position of a friendly aircraft when it detected two enemy vehicles, then the positions of the enemy vehicles would be independent.

We can take advantage of these independences by factoring our belief state. In the above example, we would write

$$P(\text{state}) = P(\text{my location}) * P(\text{enemy vehicle 1 location} \mid \text{my location}) * P(\text{enemy vehicle 2 location} \mid \text{my location})$$

Such a factorization allows us to represent each component of the belief state separately, allowing a lower-dimensional representation. The remainder of this section describes several individual advances in situational awareness and belief tracking which are based on the above intuition.

Role-based particle filters. We designed technology for allowing a particle filter to represent a distribution over a large number of enemy locations by factoring identity uncertainty away from position uncertainty. By doing so, we are able to track a belief such as "the UAV which I saw at coordinates <x,y> has now advanced to <x',y'>" even if we cannot determine whether this UAV is the same vehicle as another UAV which we saw 20 minutes earlier.

FastSLAM. FastSLAM is our name for a set of computational methods which we have designed to help make RBPFs scale to larger spaces. These methods are based on organizing the particles into a tree and updating pieces of the tree for each new observation.

Query-response model. In this algorithm for distributed situation awareness, agents maintain independent beliefs, ask neighbors for data, and pass useful data down the chain of network links in response to queries. Data which is being considered for communication is evaluated according to the metric of surprise: if the information which I have would be surprising to my neighbor, I will send it to him at the next available opportunity. We have evaluated the query-response algorithm in simulation and experiments, and determined that it provides bandwidth and scalability benefits in comparison to alternative communication rules.

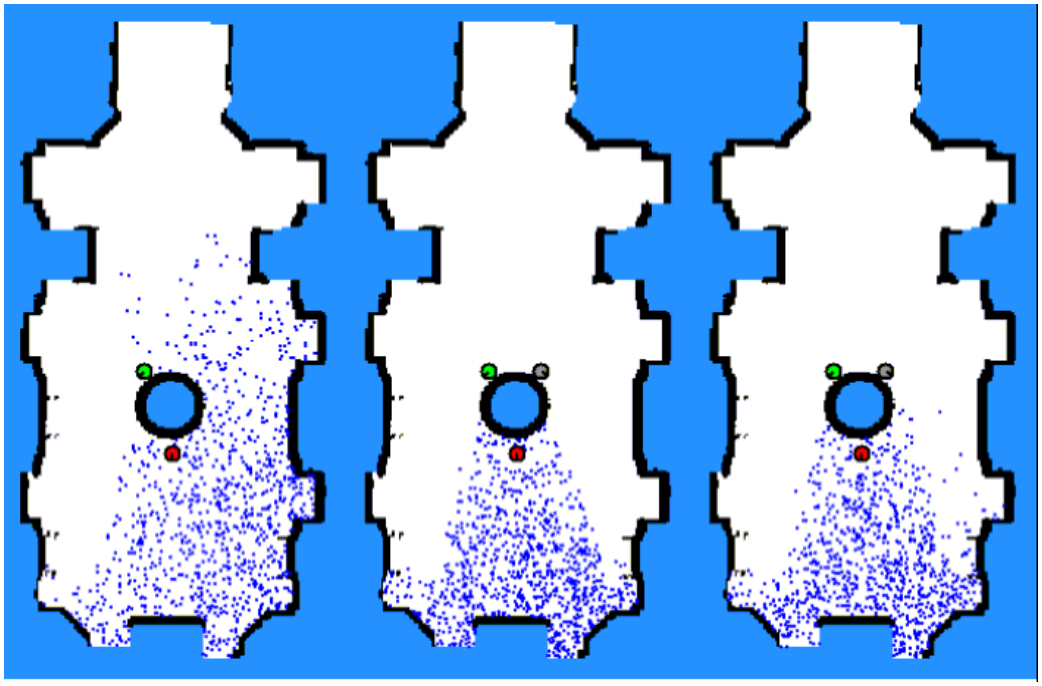

Figure 2. The query-response algorithm in action.

In the center panel, a team of two agents with unlimited communication (green and grey circles) is able to determine the correct posterior distribution (blue dots) for the location of an enemy UGV (red circle) by combining their observations: the green agent can tell that the red UGV has not passed through the left-hand corridor, while the gray agent can tell that the red UGV has not passed through the right-hand corridor. In the left panel, the green agent alone is unable to determine the correct posterior. In the right panel, by using the query-response algorithm, the green and gray agents are able to maintain an accurate approximation of the true posterior distribution even though they are only able to communicate over a limited-bandwidth link.

Intent tracking. Traditionally, belief tracking is limited to physical parameters of enemy unmanned vehicles. We have implemented a system which tracks evidence about the intent of enemy vehicles as well: by comparing observed movements to an expert-provided "playbook" of possible hostile maneuvers, we are able to predict some information about the likely future motions of the vehicles.

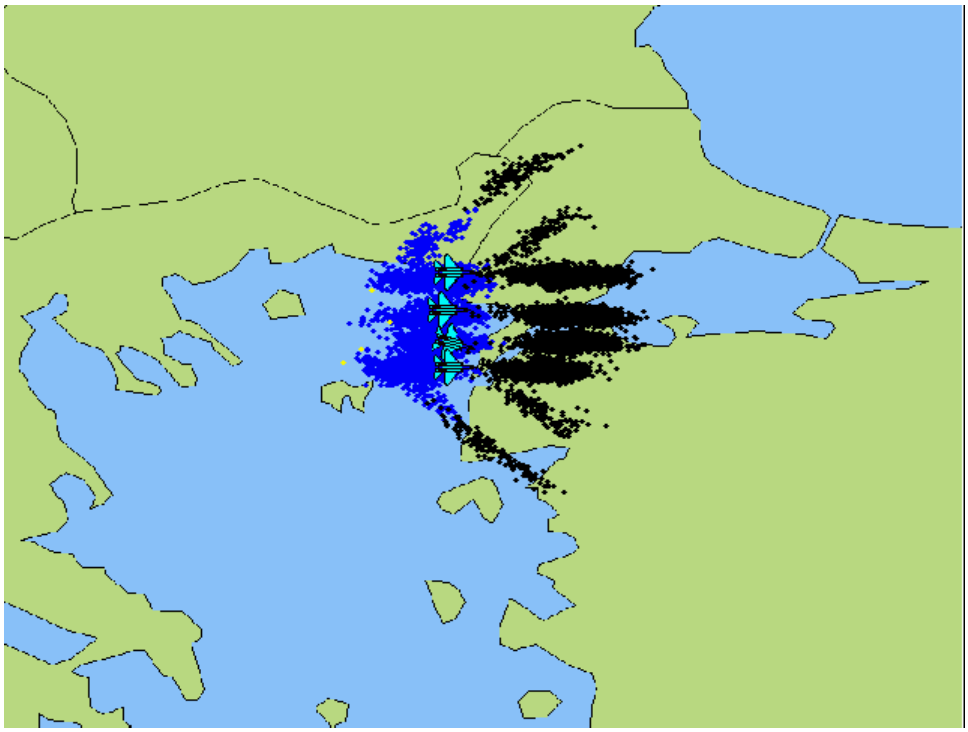

Figure 3. The intent tracker.

Four UAVs are being tracked through noisy observations. The posterior distribution for their current position is shown in blue; predicted future locations are shown in black. The intent tracker predicts that the UAVs will either continue forwards, or begin to spread out according to the maneuver shown below.

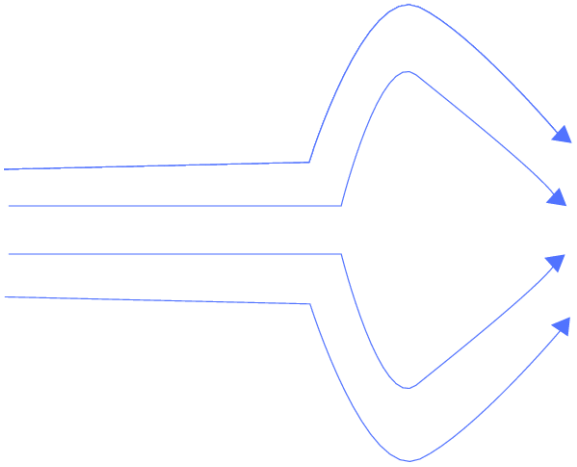

Figure 4. A sample play from the playbook: four vehicles travel in formation until reaching a waypoint near the target, then spread out to approach the target from multiple angles.

5. PLANNING UNDER UNCERTAINTY

Because the problem of planning under uncertainty for multiple agents is an active open area of research, we developed several new efficient planning algorithms for use in the MICA domain. This section describes three areas where our research achieved advances to the state of the art: belief compression for single-agent planning, what-if analysis for planning in the presence of an adversary, and market-based coordination methods for decentralized planning.

5.1 BELIEF COMPRESSION

Standard value function approaches to finding policies for Partially Observable Markov Decision Processes (POMDPs) are intractable for large models. The intractability of these algorithms is due to a great extent to their generating an optimal policy over the entire belief space. However, in real POMDP problems most belief states are unlikely, and there is a structured, low-dimensional manifold of plausible beliefs embedded in the high-dimensional belief space.

We introduced a new method for solving large-scale POMDPs by taking advantage of belief space sparsity. In this method we reduce the dimensionality of the belief space by exponential family Principal Components Analysis, which allows us to turn the sparse, high-dimensional belief space into a compact, low-dimensional

representation in terms of learned features of the belief state. We then plan directly on the low-dimensional belief features. By planning in a low-dimensional space, we can find policies for POMDPs that are orders of magnitude larger than can be handled by conventional techniques.

Large Partially Observable Markov Decision Processes (POMDPs) are generally very difficult to solve, especially with standard value iteration techniques. Maintaining a full value function over the high-dimensional belief space entails finding the expected reward of every possible belief under the optimal policy. However, in reality most POMDP policies generate only a small percentage of possible beliefs. For example, a UAV navigating in a combat theater is extremely unlikely to ever encounter a belief about its pose that resembles a checkerboard.

If the execution of a POMDP is viewed as a trajectory inside the belief space, trajectories for most large, real world POMDPs lie near low-dimensional manifolds embedded in the belief space. So, POMDP algorithms that compute a value function over the full belief space do a lot of unnecessary work. Additionally, real POMDPs frequently have the property that the belief probability distributions themselves are sparse. That is, the probability of being at most states in the world is zero. Intuitively, unmanned vehicles and other real world systems have local uncertainty (which can often be multi-modal), but rarely encounter global uncertainty. We will take advantage of these characteristics of POMDP beliefs by using a dimensionality reduction technique, exponential family Principal Components Analysis (E-PCA). In E-PCA, we employ a link function to transform the data into a space where it lies near a linear manifold. E-PCA will allow us to find manifolds with only a handful of dimensions, even for belief spaces with thousands of dimensions.

Our algorithm begins with a set of beliefs from a POMDP. It uses these beliefs to find a decomposition of belief space into a small number of belief features. Finally, it plans over a low-dimensional space by discretizing the features and using standard value iteration to find a policy over the discrete beliefs.

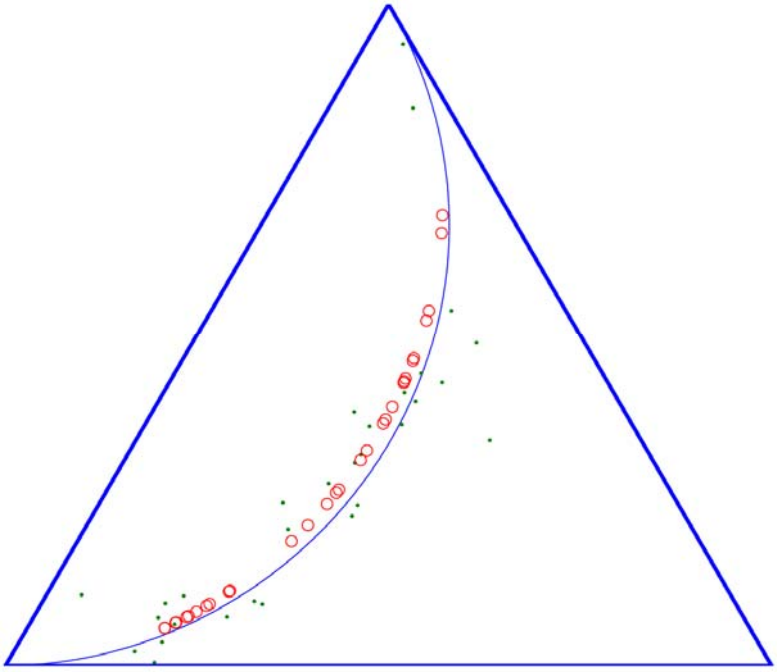

Figure 5. A schematic representation of belief compression.

The triangle represents the set of possible belief distributions in a world with 3 states. (Real problems have many thousands of states, but such a diagram would not fit on a 2-dimensional page.) The blue curve represents a lower-dimensional manifold within the set of possible beliefs. The green dots are beliefs which were encountered while exploring the world; they lie near but not on the blue curve. The red circles are the projections of the green dots onto a learned manifold; they trace out a path which lies near the blue curve, demonstrating that our learning algorithm successfully discovered the underlying structure in the data. Our planner would then use a subset of the red dots as states for its planning process.

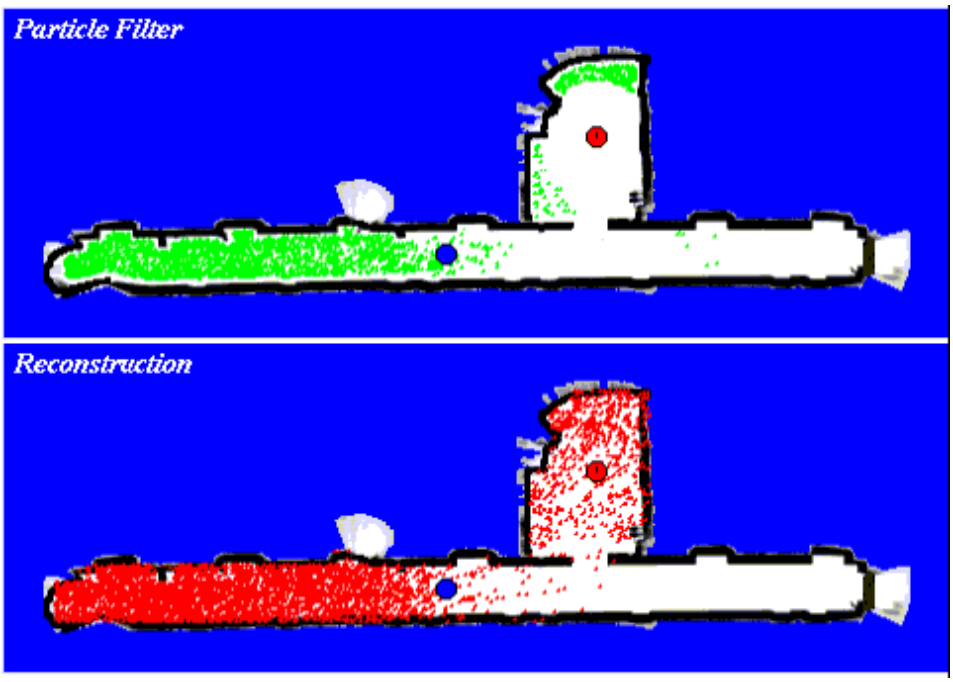

Figure 6. A snapshot of the execution process for our belief compression planner.

In the top panel is the current state of the world: our planner (the red UGV) is searching a building for the blue UGV. The true blue location is unobservable. Our belief about the blue location is shown using green dots; the entire distribution of green dots corresponds to a single data point in the previous figure. Given the current belief, our planner compresses it to a set of 6 real numbers, and uses these numbers to look up its best action in a policy table computed by the planning algorithm. The bottom panel shows the current compressed belief (red dots); it is a good (although not perfect) approximation to the true belief distribution.

5.2 MARKET-BASED COORDINATION

We have designed models for multi-agent planning problems that can be decomposed into separate single-agent MDPs which interact through the production or consumption of fictitious resources. These resources may be physical goods such as fuel; or they may be logical resources such as the right to pass over a bridge at a particular time, the right to explore an area of the environment, or the right to collect reward for achieving a particular subgoal. Time may be part of the individual robot states, in which case a

resource could be the right to consume a unit of fuel at a particular time (a futures contract).

More formally, we will phrase each individual agent's MDP as a linear program in flow form: each agent (indexed with i) has a vector of state-action visitation frequencies f_i which must satisfy its own local dynamics $A_i f_i = b_i$. These dynamics ignore any other agents in the environment. Each gent will also have a cost $c_i(s,a)$ for each state-action pair. This cost ignores any interaction with other agents. If they were to plan independently, the agents would minimize their total cost $f_i \cdot c_i$ subject to the constraints $A_i f_i = b_i$ and $f_i \geq 0$.

Sometimes, we may wish to force some of the state action visitation frequencies to take on integer values. For example, to represent a deterministic policy in a domain where time is included in the state we require that every state action visitation frequency f_{ij} be either 0 or 1.

In a multi-agent MDP where agents must coordinate, the robots are not independent and the model above is insufficient. The agents will interact through the consumption and production of resources. We define the production or consumption of resources by robot i by a matrix D_i. Element (j; k) of D_i is the amount of resource j which is produced or consumed by robot i in stateaction pair f_{ik}.

Given a description of how much of resource j each possible state action pair of every agent generates or consumes, we can write linear constraints which guarantee that production equals consumption on every time step. These constraints couple the different planning problems. We will call these kinds of constraints resource constraints.

If we combine the inter-agent resource constraints with the individual robot dynamics and individual agent objectives, we have a mixed integer program representation of the multi-agent MDP. This mixed integer program is exactly the representation that we use to represent our multi-agent coordination problem.

To solve this mixed integer program, we experimented with multiple different algorithms. All of the algorithms were based on searching for fair prices for each resource: given fair resource prices, each agent will independently plan to use its appropriate share of each resource. So, combining all of the agents' independent plans will yield a consistent global plan.

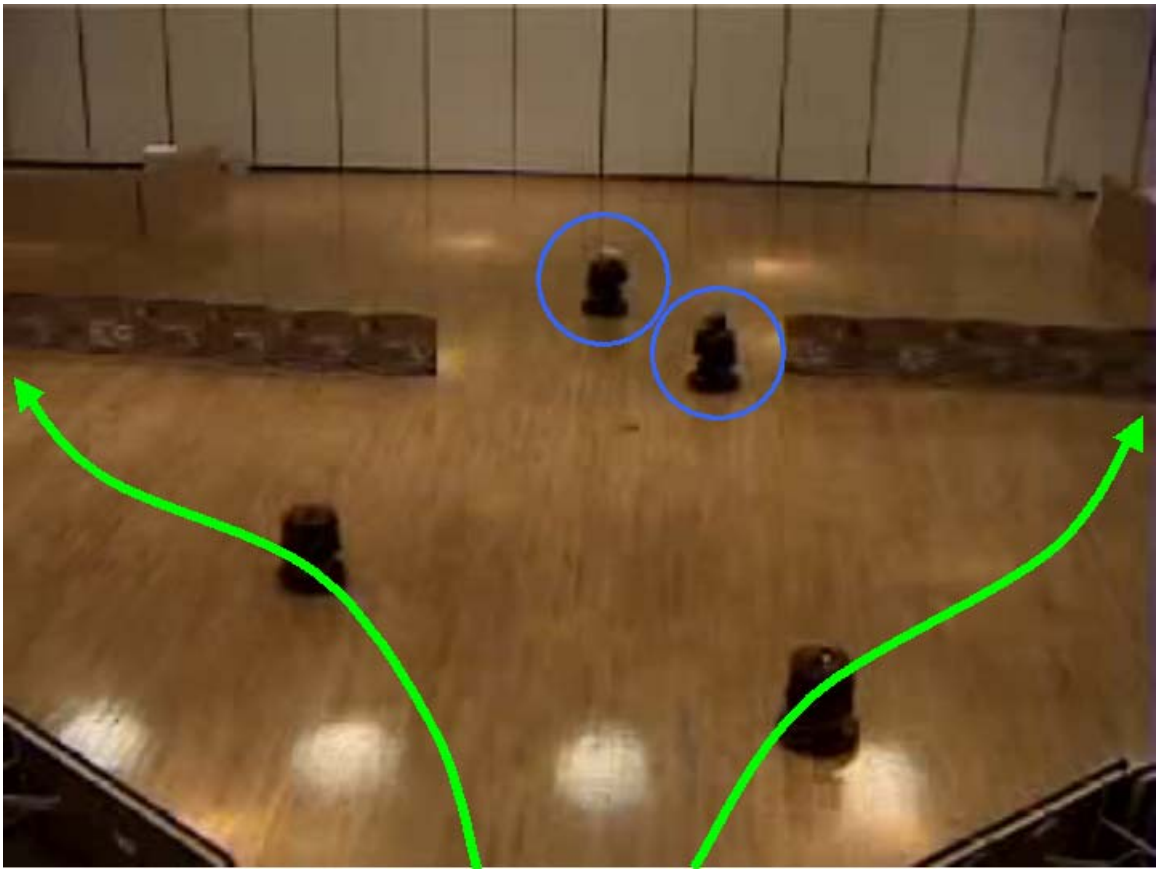

Figure 7. Combination of market-based coordination for multi-agent planning with what-if analysis for adversarial planning, as applied to our robotic testbed for the SEAD task.

(See text for details.)

5.3 WHAT-IF ANALYSIS FOR GAME-THEORETIC PLANNING

In addition to the above market-based decomposition for teams of agents into independent planning problems for each agent, we investigated how to plan when there was an adversary in the mix.

We can represent a general planning problem as a matrix game where each row corresponds to a team policy and each column corresponds to an enemy policy. This is a very large (exponential size) game, but we developed efficient algorithms to solve it by reducing it to a sequence of small linear or mixed integer programs.

One of our experimental results is illustrated in the figure: two Green team robots (representing UAVs on a SEAD mission) must plan a path from the near side of the arena to the target area on the far side. Simultaneously, two Blue team robots (representing

transporter-erector missile launchers) must decide where to set up their defensive positions. The Green robots must coordinate their paths to reach a timing point simultaneously, and must also cooperate to produce an unpredictable approach pattern to reduce the Blue team's ability to defend the target area.

In this scenario, the Green robots start by planning shortest paths from their current location to the target location. They check to make sure that the timing constraints are satisfied, and if necessary insert holding patterns to ensure coordination. Then they consider what the Blue team would do in response. In this case, the Blue team will place its defensive units in a bottleneck between obstacles (representing terrain features such as mountain ranges) along the shortest path. This position virtually guarantees them a good shot at the Green units as they approach the target area.

In response to the predicted Blue plan, the Green units devise a new plan: they will take a longer route around the Eastern side of the mountain ranges. In response to this Green plan, the Blue units can devise a new plan, and so forth.

Given a set of potential Green plans, the Green units will now determine their best **randomization** among these plans. By randomizing among plans, they make their approach less predictable, and make the Blue units' interception task harder. Within finitely many steps, our algorithm is guaranteed to converge to a minimax equilibrium of the scenario: we will determine the best possible randomized approach plan for the Green robots, and the best possible defense locations for the Blue robots.

Shown in the figure is one possible outcome of the randomization: the Green robots choose to skirt the "mountains" and approach the target area from either side, while the Blue robots have chosen to defend the central bottleneck. So, the Green team successfully meets at the target area without taking Blue fire. Other possible outcomes of the randomization include having the Blue team partially block several of the approach routes, and having the Green team attempt to fly through the central "mountain pass" area. If the Green team determines that they have chosen a route which is too well-defended, they are programmed with a reactive behavior which causes them to turn back and try again with a different plan.

6. INTEGRATION ARCHITECTURE

A substantial effort during especially the first year of the project involved establishment of a software architecture for integrating our mixed-initiative interaction framework, the POMDP hierarchy being developed by CMU that automated much of the MICA hierarchy, and the Boeing MICA OEP simulation environment we were required to use for testing. BBN designed and developed virtually all of this framework for our team's efforts.

Figures 8 shows our overall approach to the system architecture, designed to enable testing with several different operating environments, including the MICA OEP. Execution agents in this distributed agent architecture are the software agents that interact directly with the 'operational' environment, provided by the OEP in most cases. These agents direct the simulated physical agents in the OEP to move, sense, and fire weapons. These agents also have a certain amount of reactive capability to perform tactical movements in direct response to such things as radar hits they detect.

We implemented this framework using a software agent architecture based on OMAR-J that had previously been developed by BBN for several large-scale agent systems efforts for DARPA CoABS program. We used this framework to wrap the uncertainty-modeling planning and distributed information fusion algorithms developed at CMU so that they could be associated with individuals or teams of UAVs and invoked repeatedly as information was gathered from the OEP.

Frequent planning is needed as qualitative changes in the level of uncertainty occurs in a dynamic situation. For example, when new threats and targets are identified by surveillance UAVs, or when users made decisions regarding objectives and priorities, the constraints faced by the UAV team require updating, and the action policies for each team member need to be reviewed. Only when a policy already covers the contingency that is revealed through observations can the overall policy remain unchanged. That kind of uncertainty was primarily related to UAV locations, which are not very uncertain in the OEP environment (and likewise would be relatively certain in real UAVs using GPS). Similarly, since the sensor platform positions are known with high precision, opposition element locations can be determined with less uncertainty, making distributed information fusion less critical in this somewhat artificial environment. As our team's

approach to managing both predicable event uncertainty and unpredicted observation uncertainty was the cornerstone of our research effort, we designed our architecture to support several different experimental platforms to be able to effectively evaluate our work.

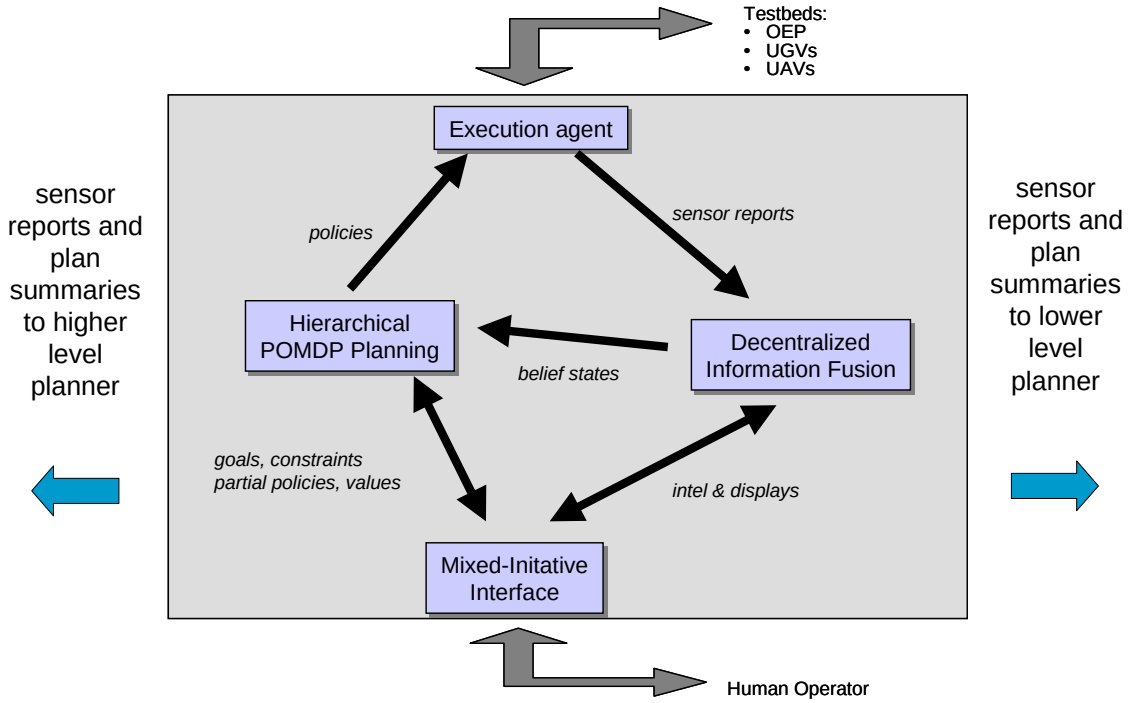

Figure 8. Architectural framework for MICA U[4] interaction with testbeds.

Sensor reports from agents with sensing capability can pool the information that they collect using distributed techniques to develop uncertain models of object locations using one of several techniques under development by the CMU and University of Washington team members.

The POMDP-based planning system uses this processed observational information in conjunction with threat and objective information provided through the mixed-initiative interface to develop or revise action policies for all agents, including possibly reassigning tasks to different agents as targets come and go, or threats change the optimal paths to objectives.

Figure 9 shows an early conception of the relationship between different agents operating both as elements of the MICA hierarchy and the console for interacting with a human operator, displaying the overall situation and current state of planning. Note that

this display is similar to (using the same mapping subsystem) but contains different information and controls than the OEP native display. Agents 2 and 3 in this figure represent individual UAVs executing missions within an overall plan developed by a team leader agent that might be associated with a surveillance aircraft. The communication between the team planner and the human operator is done using an XML –based constraint and objective description language described briefly in the next section.

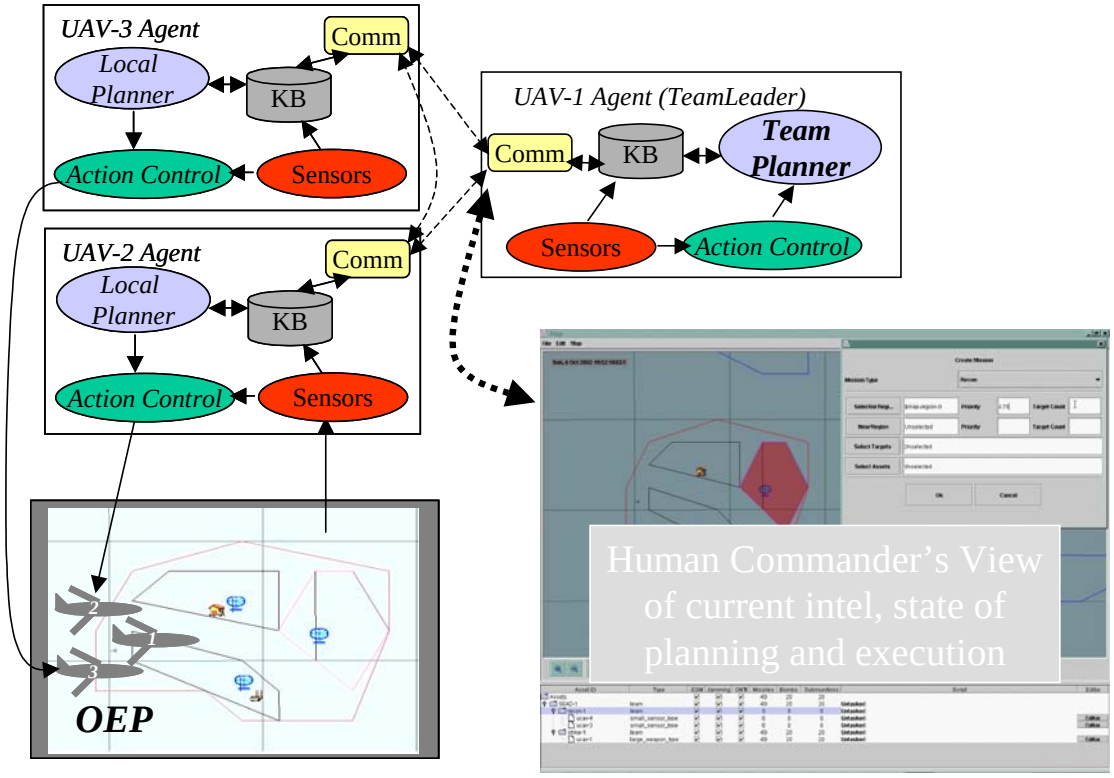

Figure 9: Relationships among agents and situational views.

6.1 PLANNING CONSTRAINT LANGUAGE

In order to support effective interactions between what is essentially a map-based interactive display and the hierarchical POMDP-based planning system, we worked closely with CMU over the first six months of the effort to design an interaction language that could be used to represent planning constraints and objectives that could be acquired using GUI tools (possibly combined with natural language interactions) and communicate them to the agents doing planning.

This inter-agent communication language expresses a combination of goal/utility information and its converse (threat/avoidance), in conjunction with spatial and temporal

relations suited to reformulation as POMDP input elements with appropriate indications of levels of uncertainty. Users can thus describe points and regions in space as

- Threats or obstacles (represented as Gausians over areas to avoid)
- Goals as points or regions in space **x** time **x** heading
- 'Corridors' assigned to sets of agents (in multi-layer POMDP policies) that introduce sequences of agent-specific subgoals with avoidance regions (called 'waypoints').

The initial version of this language addressed primarily path-planning constraints. Its elements are summarized in Figure 10. During the third year of the project we had planned to extend this formalism to address uncertainty in conjunction with more explicit coordinating temporal constraints among multiple agents.

- Positions (Points, Regions)
  - **certain (exact positions)**
  - **fuzzy (points w. variance)**
- Objects (OEP types)
- Vectors (point+heading)
- Relations
  - **At** (agent, pos|obj),
  - **Near** (agent, pos |obj)
  - **Distance-from** (obj1, obj2)
  - **On** (agent, vector)
- Actions (other than move)
  - e.g. fire weapon
- States
  - logical combinations of relations, actions
- Goals, Waypoints
  - specified as states with positive utility
  - May include action to take in goal state (e.g., fire weapon)
  - may be temporally ordered
- Threats, Obstacles
  - specified as states with negative utility

Figure 10: Elements of Constraint Language

```
<Mission ID="mission463">
        <MissionType>recon</MissionType>
        <sequence>
                <waypoint>
                        <location>...</location>
                        <variance>.0001</variance> ; degrees²                    </waypoint>
                <survey>
                        <agent><UAV ref="UAV12"/></agent>
                        <region ID="region1">
                                <corner>...</corner>
                        ...
                        </region>
                        <subject>
                                <type>Red-Unit</type>
                                <max-cnt>5</max-cnt>
                                <certainty>.75</certainty>
                        </subject>
                </survey>
        </sequence>
</Mission>
```

Figure 11. Example constraint language expression

7. MIXED-INITIATIVE INTERACTION TECHNIQUES

7.1 VISUALIZATION OF PLANNING POLICIES

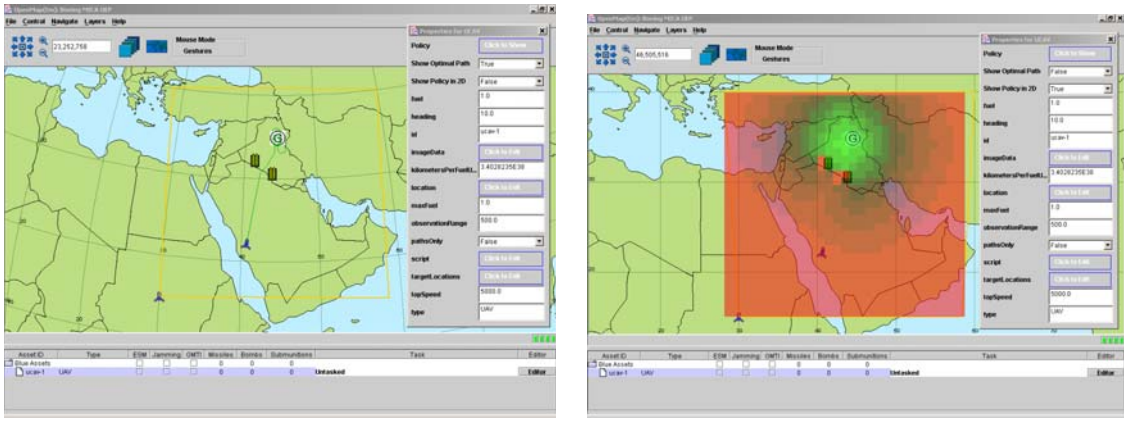

Figure 12. Year One views of single UAV (best) plan and overall policy.

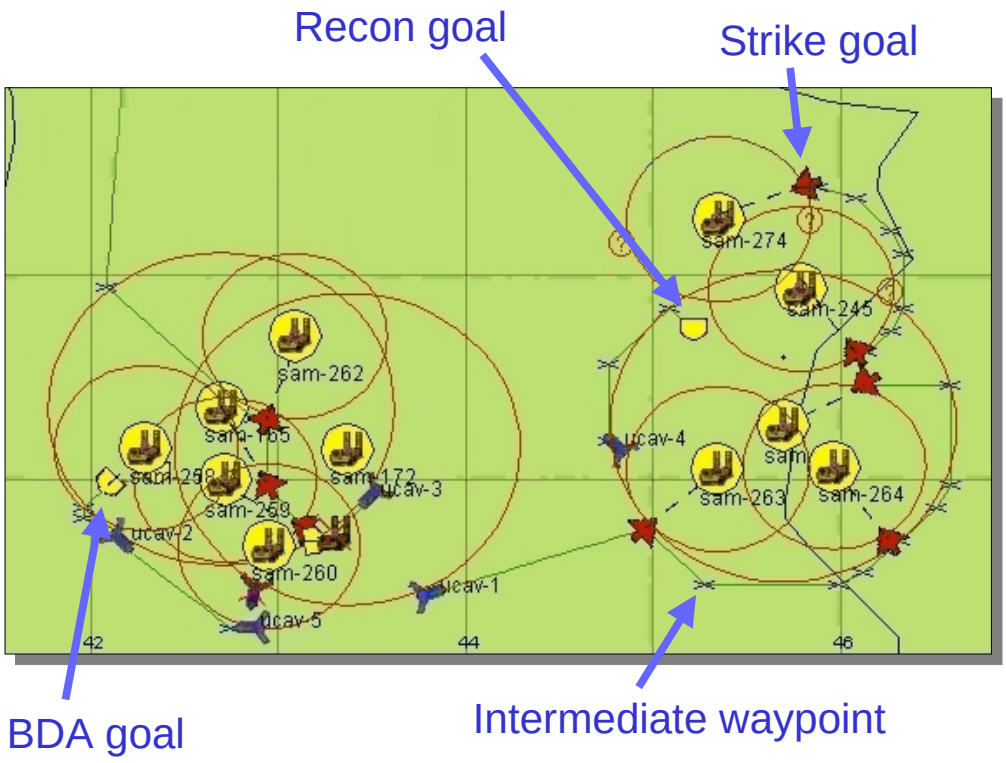

Figure 13: Iconography of Plan/Situation Visualizer from Year 2

8. REACTIVE AGENT BEHAVIORS

We provided our UAV controller agents with a suite of behaviors that improve performance on the SEAD mission's goals of destroying enemy SAM sites with minimal UAV loss. These behaviors implement high level tactics that either provide constraints to the planner, or react to events in the simulation in a pre-determined way. The commander can enable or disable these behaviors at initialization time. It is certainly possible for the commander to also turn on or off the behaviors dynamically, while the UAVs when are in flight, however the functionality for doing so hasn't yet been built into our commander's interface. We ran experiments to determine the effectiveness of these behaviors by comparing the performance metrics of the UAVs with each behavior turned on or off. The experiments were run in a Monte Carlo fashion, with 20 runs averaged for each condition. The following sections describe each behavior and its effect on UAV performance. Appendix X gives the detailed statistics gathered for these experiments.

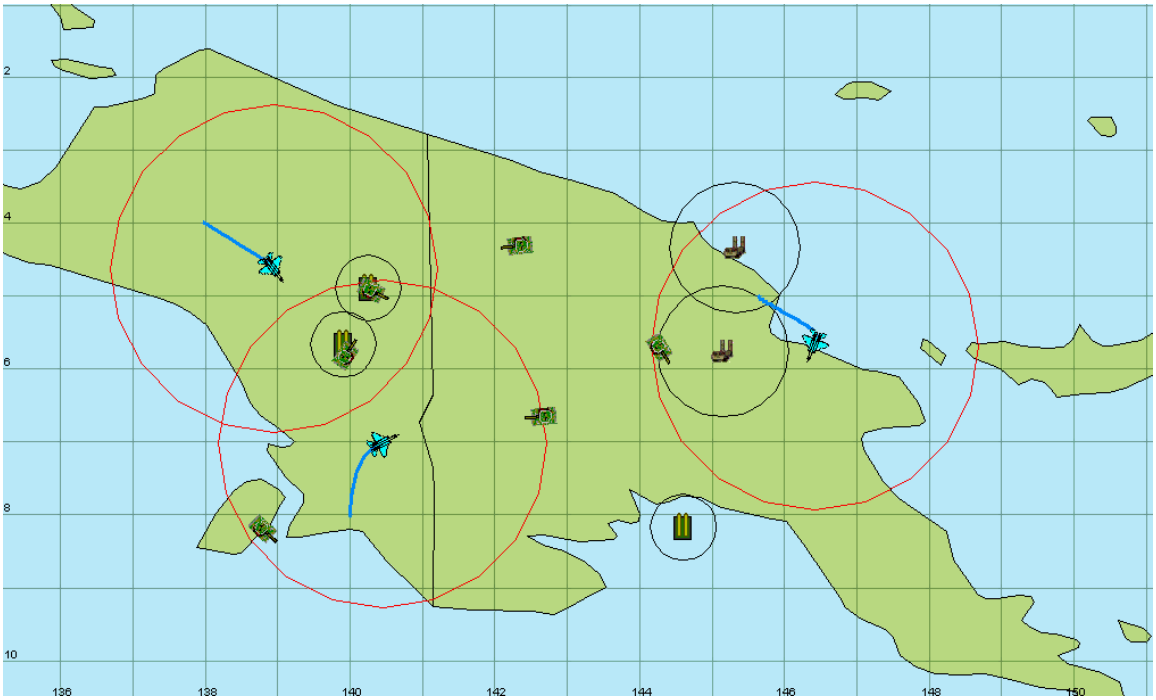

Figure 14. An example of a reactive behavior.

Reactive behaviors can range from simple to complex; in this behavior, three UAVs (with sensor ranges indicated by red circles) try to avoid threat areas (black circles) while keeping enemy UGVs under surveillance.

8.1 OEP PARAMETERS

The scenario goal for all of the experiments described in this section was to destroy all SAM Sites in Area 2, the kill region directly north of the blue base. This region contained 7 SAM sites, two long range and five medium range. At the start of the scenario, the commander knows the approximate location of two of the SAM sites, one long range, and one medium range. The remaining five SAM sites must be discovered by the UAVs. Five UAVs were assigned to the team responsible for this SEAD mission, two small weapon UAVs, two small sensor UAVs, and one small combination UAV. All UAVs had sensors capable of locating a SAM site. The weapon and combo UAVs were able to destroy SAM sites by shooting a seeker missile. The sensor UAVs were able to perform BDA (battle damage assessment) to determine if a SAM site was destroyed or not. For the performance metrics, a SAM site was only recorded as destroyed if a sensor UAV was able to perform a successful BDA, even if the site was actually destroyed in

the OEP. Therefore, if both sensor UAVs get destroyed or both weapons and the combo get destroyed, no more SAM sites will be destroyed, regardless of how long the scenario runs. The experiments ran for a set length of time, which ended up being about four hours of OEP simulated time. This length of time was large enough that the UAVs either destroyed all 7 SAM sites, or enough UAVs were destroyed, making it impossible to complete the mission.

The OEP used the 3-D radar signature for the UAVs (not the fuzz ball signature), so that UAVs moving away from or perpendicular to a tracking radar have a higher probability of being detected than a UAV moving directly at a target. Running the scenario with the fuzz ball signature instead (UAVs have a low chance of detection from all angles) made it too easy for the UAV team to achieve near perfect performance.

8.2 LONG SAM FIRST

This behavior added a simple constraint to the planning system. Any long range SAM sites must be struck before any medium or short range SAM site. Since this is implemented as a high level tactic, it is a hard constraint, not allowing the planner to plan any strike action against any medium range SAM site until all long range SAM sites have been struck. The perceived benefit for this constraint is increased survivability for the UAVs, since they are eliminating the greatest threats first. The threat areas of the SAM sites overlap enough, so that when a UAV engages a medium SAM site, it might be necessary to fly through the threat region of a long SAM site. Due to the radar signatures of the UAVs, they have a lower likelihood of getting tracked by the SAM radar while heading straight at a target, which would be the case when a UAV is striking a target, then when flying perpendicular to the target, which would usually be the case when flying through a threat region to get to another target. The downside for using this tactic would be the extra time required, when some of the UAVs that could be striking targets are forced to wait for the long SAMs to be destroyed first.

The simulation results confirm this prediction. On average, one more target was destroyed using the "Long SAM First" behavior than without it. The end time of the simulation however, was almost double. In addition, the results show that some of the other behaviors, described below, were necessary for successfully shooting down the long range SAM sites. With no other behaviors turned on, "Long SAM First" performed

much worse on than off. This is because the UAVs sent after the long range SAM sites were struck down fast. Without this behavior on, those same UAVs were able to take out a medium SAM site or two before getting destroyed.

8.3 SHOOT LONG SAM SITES TWICE

Another simple behavior, "Shoot Long SAMs twice" was an operational tactic that didn't impact the planning system. Instead of firing one missile at a long range SAM site, the UAV would fire two. This tactic makes sense, since long range SAM sites have numerous components and could still function even if some of the components were damaged or destroyed. This behavior would have more implications for the planner if we included a resource model for the missiles, and planned for returning to the base and reloading. Since the scenario we experimented with was a small subset of the full challenge problem, we were able to avoid resource usage constraints, and assume the UAVs have enough missiles to carry out their mission.

With "Shoot Long SAM sites twice" and "Long SAM First" turned on, the UAVs were able to destroy 2 more SAM Sites on average. This is because the strike UAVs after the long range SAM sites have a higher probability of kill, and thus can survive longer to strike other targets. Without "Long SAM First", the "Shoot Long SAM sites twice" behavior didn't make much of a difference, because most of the SAMs destroyed were the medium range ones.

8.4 FACE THREATS

This is a reactive behavior that was triggered whenever a UAV's threat warning sensor indicated that a SAM radar was locked on. Since a UAV presents a smaller radar signature when pointed directly at a hostile radar site, flying towards a threatening radar may be a better tactic than turning around and presenting a larger radar cross section while trying to leave the threat region. This is an example of a type of behavior where it would be interesting to have the commander dynamically add the behavior after noticing UAVs getting shot down when they turn to flee a threat region. Our current commander's interface does not support the capability to dynamically add a behavior.

This behavior had the largest effect with "Long SAM First" turned on. Three more SAM sites were destroyed on average with "Face Threats" on. Even with "Shoot Long SAMs twice" turned off, "Face Threats" allowed the UAVs to shoot down two more

SAM sites on average. The effect is less pronounced when engaging medium range SAM sites, because it the time a UAV needs to travel inside the threat region of a medium SAM is much lower than for a long SAM.

8.5 DIVE WHEN THREATENED

Another simple reactive behavior, the "Dive when Threatened" tactic causes the UAVs to decrease altitude whenever their threat warning radar is active. Tactically, a downside to this behavior is that a lower altitude opens the UAV's up to hand-launched AAA weapons, which can strike without warning. Strategically, a lower altitude limits the sensor range of the UAVs, lowering the chance of detecting a target from a longer range. After the threat warning sensor turns off, this behavior causes the UAVs to return to their cruising altitude (chosen by the commander), which limits the strategic downside. Since this behavior only turns on when the UAV is in direct danger of being shot down by a hostile SAM site, the added danger of small arms fire is not too important, since the likelihood of a kill by small arms fire is low.

Most of the experiments were done with "Dive when Threatened" turned on. One set of runs varying the state of "Dive when Threatened" showed the expected performance improvement. With the behavior active with "Long SAMs first" on, and "Shoot Long SAM sites twice" off, the UAVs were able to destroy two more SAM sites on average when with "Dive when Threatened" off. When striking medium range SAM sites, the effect was much less noticeable, but there still seemed to be a performance improvement with this behavior turned on.

8.6 LOITER CENTER

One of the latest versions of the planner allowed for the inclusion of a Loiter goal. A UAV with no tasks to do could loiter near the Loiter goal location, instead of just loitering wherever it happens to be. The newer version of the planner was smart enough to pick a spot to loiter as close to the Loiter goal location as possible without being inside a known threat region. Without this tactic, the UAVs would just loiter wherever they happened to finish their last task, which could be inside an active threat region. Since there is no downside to using this behavior, the results should show an improvement in performance. With a Loiter goal added, and "Shoot Long SAMs first" turned off, the UAVs did much better, shooting down most of the SAM sites. With this behavior, when

a UAV strikes a medium SAM Site in a threat region of a long range SAM site, it is usually able to get out of danger before the long SAM can lock on and fire. With "Shoot Long SAMs first" turned on, the results showed a slight decrease in the number of SAM sites destroyed. The high standard deviation and low number of runs points to the possibility that this result was a statistical anomaly. In addition, the average number of UAVs destroyed was lower by almost one whole UAV.

8.7 PLANNER PARAMETER EXPERIMENTS

The utility function used by the CMU planner to determine the value of a plan had a number of parameters that affected mission performance. We tuned those parameters by running experiments in a similar manner as for the behaviors, with all behaviors except "Loiter Center" turned on. The utility function is controlled by three weights, Maximum Length, Sum Squared Weight, and Total Cost Weight. The utility function calculates the maximum length of any one UAV's plan, the sum of the squares of the length of all UAVs plans, and the total length of all UAV plans added together, and multiplies each value by the appropriate factor, after normalization. These three weights must sum to 1.0, which leaves two free parameters. We hand picked ten sets of values for these parameters and ran five trials with each set.

A value of 0.8 for Max Length, 0.2 for Total Cost, and 0 for Sum Squares yielded near perfect performance. Decreasing the Max Length weight and increasing the Total Cost weight led to a decrease in the number of SAM sites destroyed, since more UAVs were getting shot down before they could complete their missions. Increasing the Sum Squared parameter also led to a decrease in performance, with a drastically increased replanning time. With Sum Squares at 0.0, the replan time was around 200 milliseconds, and with Sum Squared closer to 1.0, the replan time was over 2 seconds.

In addition, we experimented with the obstacle cost parameter. A higher value for the obstacle cost would cause the route planner to avoid threat regions more diligently, flying a longer path to avoid entering a threat region. A value of 10.0 for obstacle cost meant the planner would go ten units of distance out of the way to avoid flying in a threat region for one unit of distance. The standard setting for obstacle cost was 10.0. We experimented with values of 1, 4, and 40, with varying values for the utility function parameters. Predictably, a value of 1 led to the worst performance, since the UAVs

basically ignored the threat regions and got shot down fairly fast. There didn't seem to be much of a difference between 4 and 40, and only a minor decrease in performance from 10.

9. ROBOTIC TESTBED FOR SEAD MISSIONS

In addition to the OEP testbed described above, our team used a group of physical robots to evaluate some of our algorithms for reasoning and planning. There were two main reasons for the use of this alternative testbed: first, it forced us to design our algorithms so that they could be interfaced to a real physical system (as well as to a software system like the OEP). This consideration required us to design for real-time efficiency as well as for real observability constraints. And second, it forced us to design our algorithms to handle the sorts of uncertainty that arise from real physical sensors, in addition to the OEP's simulated sensors. Since the OEP's sensor simulations were designed to be qualitatively rather than quantitatively accurate, we felt that it was important perform additional tests on this physical testbed.

The additional tests we performed were in the context of a suppression of enemy air defenses (SEAD) task: some robots became simulated UAVs, while others remained in a fixed location to simulate air defense units such as missile launchers. A typical scenario would require the UAV team to decide on an approach path to a given target area, while the defense team would have to deploy its missile launchers to attempt to block the most likely approach paths. As shown in the figure, the UAV team was able to plan to randomize their approach paths to avoid air defense units, and was able to use reactive behaviors to quickly alter their paths in case they ran across an unexpected hazard.

Figure 15. Two of the robots used in our SEAD testbed.

They are approximately 30cm in diameter, and can travel at 2 m/s and rotate at 10 rad/s. Their primary sensor is a SICK laser rangefinder. We have 7 of these robots, of which up to 4 at a time were used in the evaluations.