Autonomous Robots 5, 97–110 (1998)

Integrated Premission Planning and Execution for Unmanned Ground Vehicles

EDMUND H. DURFEE, PATRICK G. KENNY AND KARL C. KLUGE
EECS Department, University of Michigan, Ann Arbor, MI 48109
durfee@umich.edu
pkenny@umich.edu
kckluge@umich.edu

Abstract. Fielding robots in complex applications can stress the human operators responsible for supervising them, particularly because the operators might understand the applications but not the details of the robots. Our answer to this problem has been to insert agent technology between the operator and the robotic platforms. In this paper, we motivate the challenges in defining, developing, and deploying the agent technology that provides the glue in the application of tasking unmanned ground vehicles in a military setting. We describe how a particular suite of architectural components serves equally well to support the interactions between the operator, planning agents, and robotic agents, and how our approach allows autonomy during planning and execution of a mission to be allocated among the human and artificial agents. Our implementation and demonstrations (in real robots and simulations) of our multi-agent system shows significant promise for the integration of unmanned vehicles in military applications.

Keywords: cooperating robots, coordination, multi-agent system

Introduction

Keeping people out of harm's way is a driving motivation for the development of unmanned ground vehicles (UGVs). UGVs (Fig. 1) can operate in hazardous environments, such as in contaminated areas or behind enemy lines on battlefields, conducting reconnaissance, surveillance, and target acquisition (RSTA) on behalf of distant human operators. Specifically, in this paper, we will consider the development of UGVs for scouting missions in military settings.

Fielding UGVs for scouting missions means that the UGVs will be placed under the control not of roboticists, but of soldiers who are commanding a variety of human and mechanized systems. This imposes significant constraints on the development of UGVs and the tools for their control. First, because an operator is a soldier, UGVs need to be tasked in military terms, similarly to how manned scouting vehicles would be tasked, rather than in robotic terms. Second, because an operator is simultaneously supervising various systems, the operator's attention cannot be devoted to teleoperating the UGVs; instead, the UGVs need to be able to operate with minimal intervention by the operator. Third, because lines of communication are uncertain, an operator is only sporadically available, and cannot be depended upon for prompt supervision and redirection.

These factors motivate the development of UGVs, and operator interfaces, that internalize substantial military knowledge, including standard procedures for achieving objectives, and that know the current objectives of an operator. With this knowledge, the overall unmanned system can be migrated from a "man-must-be-in-the-loop" to a "man-can-be-in-the-loop" strategy, such that robust and adaptive operation is not entirely dependent on the man-in-the-loop.

Toward this end, our research has treated UGVs as semi-autonomous agents that are capable of sensing their environment and adopting plans from their repertoire that are most suitable for achieving their objectives given the circumstances. Because the UGVs are often

*A preliminary version of this paper was presented at the First International Conference on Autonomous Agents in February 1997.

Figure 1. UGVs.

tasked in teams, our work has emphasized techniques for improving a UGV's awareness of the more global circumstances and for coordinating the plans of UGVs. Moreover, because they are also teamed with a human, the UGVs need to treat the human (or, more properly, the human-interface captured in the operator workstation (OWS)) as an agent as well, which also is adopting plans (mission specifications) from its repertoire in response to input from the UGVs and from the operator.

Rather than treat these loci of activity in a piecemeal fashion, the approach we outline here asserts that permeating all of these tasks, ranging from the specification of missions down to the invocation of robotic actions, is knowledge about how to pursue these tasks that can be combined in flexible, situation specific ways. Thus, in this paper, we describe how a procedural encoding of domain knowledge and mechanisms to use it provide a unifying framework for accomplishing the goals of (1) premission planning, (2) contingency preplanning, (3) reactive doctrinally correct response to unanticipated events, and (4) on-the-fly semi-autonomous coordination and replanning.

We have realized these goals by adapting procedural reasoning techniques (Georgeff and Lansky, 1986), embodied in the University of Michigan Procedural Reasoning System (UMPRS), to the planning and control of activities within, and across, UGVs and the operator (Lee et al., 1994). UMPRS controls a UGV based on its sensed situation and on the directives and objectives handed down from the operator's interface agent. UMPRS also controls the OWS by retrieving mission plans and taking actions to prompt the operator for refinements to objectives and mission parameters. Thus, the military intent and knowledge is distributed across the operator, the OWS, and the UGVs, and each of these can prompt another to allow mixed-initiative development of situation-specific mission definition, elaboration, planning, and revision.

In this paper, we outline the development constraints that we have faced in generating our agent processes for the UGV domain, and then describe our solution to the problems in terms of supporting premission

plan definition and elaboration, plan execution across multiple agents, and runtime replanning during mission execution. Our overall solution comprises an integrated mission planning and execution system that is inherently distributed among multiple agents that embody explicit goals, plans, and beliefs. We also evaluate our solution both using a real robotic vehicle in a limited setting, and using a standard simulation system for demonstrating all of the components of our system working together for the planning and control of four UGVs by a single operator.

Problem Definition

The challenges in deploying UGVs in military settings fall into two broad categories: sociological and technological. From a sociological viewpoint, gaining acceptance for advanced technologies, and in particular technologies that strive for some level of autonomy, is problematic in a rigidly defined community such as the military. Throughout the UGV project, reactions by military personnel to the idea of UGVs that could improvise (within doctrine) so as to persistently achieve objectives have been in a bimodal distribution centered on "enthused" on the one hand and "horrified" on the other, with few moderate opinions between. The conservative default has been to strongly rely on the operator, and to provide vehicles with enough flexibility to dodge obstacles, but not to decide on new destinations, and certainly not to fire a weapon! Introducing agent technology, which inherently makes agents more flexible and persistent in achieving their goals, therefore has been an uphill battle.

From a technological viewpoint, several roadblocks have stood in the way of developing deployable UGVs. One is the limited sensing and mobility capabilities of real unmanned vehicles. Over the last several years, significant strides have been made in terms of mobility, such that a UGV can move among destinations off-road, at least in a relatively benign environment (without trenches, wire fences, etc.). Perception has lagged behind to an extent, not surprisingly. That is, mobility concentrates perception on aspects of the world that impact traversability; more generally, situation awareness requires perception that is much broader and richer. From the perspective of autonomous UGVs, rich perception is critical, since most (if not all) autonomy is manifested in terms of responding competently to unanticipated situations.

A second technological roadblock has been the constraints imposed by legacy software and hardware components. After substantial investment in robotic control systems and user interface tools, a project sponsor is anxious to build from these, even if it ties the hands of the current developers. Thus, to a significant extent, the development of agent technologies to be embedded in UGVs and in the OWS was circumscribed by having to weave together capabilities inherent in the legacy systems.

In summary, then, the problem of developing autonomous UGVs for military use can be viewed as generating decentralized agent technologies that could interface to military users on one end and robotic vehicles on the other such that the degree of autonomous control possessed by an agent (operator, OWS, UGV) can be flexibly changed as attitudes and acceptance changes. This is summarized in Fig. 2.

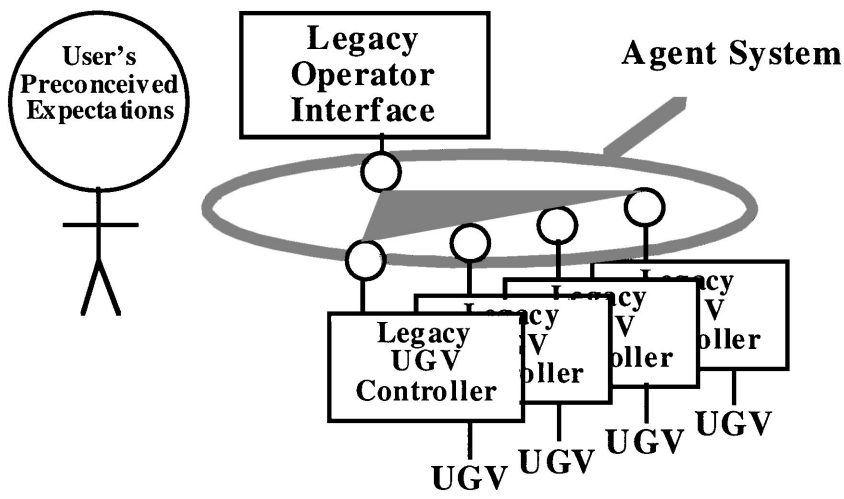


Figure 2. Agent system bridges gap.

Premission Planning

The first hurdle to overcome in bringing robotic technology into a military arena is to allow people who think in military terms to define a mission for the robotic systems without having to understand details of how the robots work. The premission planning capability supports the operator by assisting, and sometimes automating, the translation and elaboration of military objectives into robotic actions.

At the heart of premission planning is the UMPRS agent architecture, a C++ implementation of many of SRI's procedural reasoning concepts (Georgeff and Lansky, 1986), tailored specifically to support the development of agent technologies for operator assistance and autonomous execution. UMPRS supports specification of a mission's "operation order" in terms of a sequence of military procedures, each performed within the context of a larger set of objectives and constraints. Thus, whereas the robotic plan executed directly on the vehicles consists of a sequence of primitive commands, it is within the UMPRS system that the actual intent of the mission, along with the expected (and possibly alternative) strategies for its accomplishment, are represented. It is because of this more complete knowledge that the UMPRS-based premission planner is able to assist the operator: decomposing the mission into more detailed segments, possibly formulating some portions of actual robotic plans, making suggestions to the operator about features of the mission plan, and prompting the operator for further input to resolve ambiguities in the specification.

The premission planner works by accepting as input a map, generated by the (legacy) mission specification interface, that contains icons of the military measures of interest for the mission. The icons could have been entered in an arbitrary order, and the sequence and strategies for pursuing the measures might not be obvious strictly from the map. As a result, in order to resolve ambiguities the premission planner draws on its library of templates for operations, or can call on the operator to provide such a template. By matching the military symbology to the template, the premission planner begins formulating the sequence of military actions that must be carried out by the robotic vehicles. If matches are ambiguous, the interface can confirm its interpretation of the mission with the operator. If matches are impossible, the interface can prompt the operator to supply additional mission control measures, or can indicate to the operator why existing measures are inappropriate for the mission. It is important to note, moreover, that the procedures by which the mission planner's interface interacts with the operator are also explicitly represented and executed, meaning that the interface can be tailored to the needs and preferences of different operators.

As it elaborates the mission, the planner can call on more specialized tools to formulate portions of the robotic plan. These include tools for route planning (Stentz, 1995), for planning RSTA observation points (Cook et al., 1996; Kluge et al., 1994), and for planning formations (Balch and Arkin, 1995). The premission planner incorporates the information returned by these tools into its representation of the sequence of plan steps.

The final sequence of plan steps formulated by the premission planner can entirely dictate the control actions that the UGVs will invoke throughout the mission. These can be displayed to the operator (if the operator can understand them), and can be directly edited by the operator. So, in the case where the automated system is given no autonomy, the operator has the final say about precisely the robotic actions to be taken by the UGVs. If the operator relinquishes this control, the premission plan at the robotic level can be downloaded to the UGVs to be executed verbatim, meaning that autonomy is restricted to the operator and premission planner (on the OWS). However, robust performance needs dictate that the premission planner only elaborate the plan steps to an intermediate level of detail, and pass the resulting sequence of subgoals (with constraints on their accomplishment) to the UGVs, which will each then elaborate further to develop the detailed control plan. Our architecture makes no commitment to which of these different degrees of autonomy is instantiated at any given time; rather, it will depend on the particular knowledge and procedures provided to the various agents.

Contingency Preplanning

The intrinsic limitations in the accuracy with which actions can be carried out in the world, and the fact that the environment is likely to differ from what is expected, make it unlikely that a robotic vehicle can flawlessly execute a preplanned mission without complications. As a result, the UGV should be able to recover from deviations from its expectations in a variety of ways. Some deviations might involve straightforward

recovery actions; for example, if a road is barricaded, the vehicle might go offroad around the barricade and then return to the road and its original plan. Other deviations might require some immediate response; for example, if fired upon, the UGV should move to cover reflexively. But other deviations might require more deliberation; for example, if it finds that a bridge it was to traverse is gone, a vehicle would need to consider the heightened likelihood of enemy contact near such an obstacle to navigation, alternative routes to reach its original objective, and the degree to which it can still achieve its mission goals by taking those routes.

Contingency preplanning deals with the more reflexive responses that a vehicle might take, while more involved changes in plans is in the province of replanning techniques. The idea behind contingency preplanning is straightforward: before the UGV begins executing its plan, the planner should anticipate what might go wrong from a tactical standpoint, and should preprogram responses for those deviations so that the vehicle can recover without hesitation. The challenges in contingency preplanning, therefore, are in developing mechanisms by which the mission planner can anticipate likely deviations, and can plan responses in a "what if" type of mode.

We have operationalized these mechanisms as part of the UMPRS mission planning capability by extending UMPRS's representations of standard operating procedures to model the expected/intended effects of such procedures. With this additional information, we can run UMPRS in a "forward simulation" mode to capture the expected sequences of actions that will occur so as to evaluate a possible ("what if") course of action before it is actually committed to. These same capabilities support the anticipation of contingencies, since the predicted results of a procedure can also capture the procedure's possible faults or side effects.

Moreover, we can associate particular contingencies with particular primitive procedures. For example, a common contingency in following a road might be that the road is obstructed; thus, whenever the planner inserts a step to follow a road, it automatically includes the contingency plan for dealing with an obstructed road. In fact, possible contingencies can be associated not only with primitive plan steps, but also for more extensive steps, such as the contingency of receiving fire that could be associated with a larger sequence of operations behind enemy lines.

Having identified a deviated state into which a vehicle might be thrown, the planner must generate a robotic plan for responding to that state. Thanks to the fact that the intent of the mission is captured in the UMPRS-based planner, the contingency preplanning capabilities can search for procedures to invoke that robustly pursue the mission intent, including actions that remove the deviation so that the original plan can be resumed. Because the intent also can capture high-level goals (such as survival), the contingency preplanner can also suggest responses that abort portions of a mission in favor of these other goals.

Contingency plans are elaborated by the same mechanisms as premission planning, including prompting the operator for information if needed (such as "If attacked, where should the vehicles fall back to?") and invoking more specialized tools (such as to determine a formation to use if surprised by an enemy). The contingency plans are woven into the original mission plan as conditional branches.

Vehicular Planning and Execution

Depending on the level of autonomy assigned to the vehicles, they might engage in further premission planning prior to executing the mission. As discussed above, at the premission phase, the operator works with the interface provided by the OWS to specify and elaborate the mission. It could be that the OWS serves only as an editing tool, such that the entire robotic mission is specified directly by the operator. Or the operator and OWS could cooperatively formulate the detailed robotic plan. In such cases, a UGV simply receives a sequence of commands in its (legacy) robotic control language, and blindly executes these until done or until interrupted from above.

Our architecture assigns a UMPRS agent process to each UGV, which means that, in principle, a UGV can elaborate plans both before and during a mission. In the degenerate case, the UGV's UMPRS agent process maintains a single goal of passing the low-level sequence received from above on to the robotic control system. More generally, however, our assumption is that the operator and OWS will *not* elaborate missions into completely executable robotic plans, both because of the centralized computational burden this imposes at the operator level (failing to take advantage of inherent parallelism), and because of the lack of awareness and flexibility that this imposes on the vehicles. Therefore, the model we generally assume is that the operator and OWS elaborate the mission down to the level where

coordinated subgoals can be distributed to the vehicles, and it is up to each vehicle to decide how best to accomplish the objectives that it is handed. The vehicles are thus participants in the premission planning phase, elaborating a sequence of subgoals received from the OWS into detailed robotic control commands, in precisely the same way that UMPRS elaborates goals into more detailed steps at the OWS level.

A clear advantage of this strategy is that the UMPRS mechanisms for plan elaboration are inherently responsive to changing circumstances. When deciding what primitive action to take next, a vehicle will consider alternative ways of accomplishing its next goal, taking into account the current context in which it is operating. As context changes, details of how goals are being accomplished will change (different procedures will be retrieved from the library of standard operating procedures), within the more persistent framework of higher-level procedures that are still being adhered to.

The context in which a vehicle operates includes its awareness of the external environment, the internal status, and the activities of other vehicles. Information from all of these sources must be assimilated to classify the current operating conditions (Kluge et al., 1994). In our architecture, this assimilation is done through fusion processes that act on information stored in the Information Assimilation DataBase (IADB). The IADB uses a CODGER-like blackboard (as developed for autonomous land vehicles at CMU (Stentz, 1990)) to store the information about the world used by the UGV agent process. The information is collected from UGV sensors (e.g., camera), from internal monitors (e.g., fuel gauge), and from communications systems (e.g., radio), and combined into symbolic assessments of the situation (e.g., enemy-nearby? is true). To establish context for invoking a procedure, the UMPRS process can query the IADB. To ensure that a procedure remains valid, the UMPRS process can pose a standing query to the IADB, such that the IADB will respond when the conditions of the query are met. And, in some cases, a query from the UMPRS process can trigger the IADB to return goals back to UMPRS. For example, if UMPRS wants to know whether a route is safe, the IADB might trigger observation point planning, essentially saying that it can answer the question if the vehicle carries out a sequence of observations.

Finally, because we generally assume that missions are being carried out by multiple UGVs, a critical component of context is the status of other friendly vehicles. Many cooperative procedures embed communicative actions for maintaining this context, such as in a "bounding overwatch" maneuver where two vehicles leapfrog through an area, taking turns watching over the other and then being watched over in turn, where the turn taking is often synchronized through messages. However, in cases where messages fail to materialize, or where vehicles should maintain radio silence, the vehicles can use observations made with their sensors to draw inferences about the status of other vehicles (including enemy vehicles) (Huber et al., 1994).

Runtime Replanning

Treating individual vehicles and the user interface process at the OWS each as agents provides considerable flexibility not only to premission planning and flexible execution, but also to how vehicles should respond to deviations between expected and actual circumstances during plan execution. In some cases, simple feedback control mechanisms might suffice, such as swerving to avoid an obstacle. In other cases, plans for contingencies might have already been developed, such as fording a stream when a vehicle discovers that a bridge has been demolished.

However, at times more wholesale replanning might be called for. For example, if the vehicles were intended to perform a triangulated reconnaissance to localize some enemy force, the loss of some of the vehicles could jeopardize the mission. A major plan change might be needed, such as distributing the vehicles in radically new locations to approximate the localization effort, or rapidly moving vehicles among multiple points so that each is making several observations, or aborting the initial objective and pursuing a secondary mission goal.

Several replanning strategies exist. Again, at the most degenerate level, sensory data collected by the vehicles can be directly passed to the OWS and simply displayed to the operator, who is responsible for monitoring plan progress and adjusting the detailed robotic plan as needed (that is, when circumstances invalidate the current plans, including contingency plans downloaded to the vehicles). This gives the operator the greatest "hands-on" control, at the cost of requiring the operator to dedicate complete attention to the UGVs.

Because the OWS is an agent, with the associated running UMPRS process and IADB, it can be allowed to perform replanning under some circumstances. In this case, the vehicles return data to the OWS, which

the OWS processes (adds to the IADB, where fusion/interpretation processes can act on it). The OWS is internally "executing" the mission plan, following the progress of the vehicles and updating the operator's map, and so can use the IADB updates to check the context of (components of) the mission plan to identify violations of context. These violations, in turn, trigger the OWS to retrieve alternative procedures for accomplishing goals that are valid for the context, elaborating suitable procedures (potentially requesting input from the operator), and sending new instructions down to the vehicles to supersede the vehicles' current sequences of commands. Because the OWS internalizes knowledge of the mission intent and methods for accomplishing that intent, along with knowledge of what might invalidate the premission plan and contingency plans, it can automate recovery replanning in this way.

Of course, to the extent that the UMPRS agent processes on board the vehicles are themselves tasked in terms of objectives and are knowledgeable about alternative procedures, the vehicles can similarly detect plan deviations and replan. In our system, each vehicle can use its own local IADB to update its knowledge about the context. Context changes trigger the retrieval of alternative plans for achieving the vehicle's goals. In addition, because a vehicle maintains top-level goals including keeping other vehicles and the OWS informed of significant context changes, it volunteers some of the contents of its IADB to others, such that their IADBs are updated, triggering further changes either at other vehicles or across the mission at the OWS. In fact, these top-level goals inherently are geared toward coordination, and can trigger cooperative replanning among the vehicles, should they detect that their current plans are outdated and that the OWS is unable (not responding) to oversee their collective replanned activity.

When vehicles perform local replanning, however, it opens the door for uncoordinated activity since their local responses might fail to achieve a synergistic whole. Or worse, their local responses might be conflicting. To avoid this, it is important that the range of replanning choices available to the vehicles be limited enough to avoid conflict, without being so limited as to unnecessarily constrain the vehicles. General constraints, such as so-called "social laws" can serve this purpose (Shoham and Tennenholz, 1994), but often impose constraints for situations that might be impossible given the plans already adopted by the vehicles. An alternative is to allow the vehicles to exchange explicit information about their revised plans, and to search for a minimal set of commitments that assure conflict-free execution (Lee, 1996).

Implementation and Evaluation

The component technologies to build the interacting agents that bridge the gap between the operator interface and the robotic vehicles, as outlined above, have been implemented and selectively deployed. UMPRS has been implemented as the backbone of each of our agents, both for on-board vehicle planning and execution, and for controlling the functionality of the OWS. For mission planning, UMPRS has interfaced to planning tools for formation planning, route planning, and observation point planning. The IADB has been implemented, and its interface with UMPRS has been built. At present, sensor fusion techniques associated with the IADB are for the most part primitive, such that subtle context changes are not recognized, but blatant ones (such as the appearance of an enemy) are captured. The most advanced technique we have implemented for assimilating information has been geared toward plan recognition using belief networks (Huber et al., 1994; Huber, 1996).

Bringing all of these technologies together for an overall evaluation using real UGVs has proven a challenge, primarily because of the inherent limitations, scarcity, and unreliability of current experimental UGV platforms. As a result, we have tested smaller combinations of technologies on various simpler robotic platforms, and a larger combination of technologies within a realistic, widely used simulation system. In what follows, we summarize our observations.

Reactive Execution in an Outdoor Mobile Robot

We have used UMPRS as a controlling agent process for an outdoor electric utility vehicle (Lee et al., 1994). The vehicle navigation system included the YARF road follower (Kluge and Thorpe, 1995) to allow the vehicle to navigate along roads and sidewalks. It was also equipped with the ability to detect a simple class of landmarks (orange traffic cones), and to drive the vehicle off-road to a specified landmark in the environment.

In our experiments the robot's mission was to carry supplies along a road previously scouted by another vehicle. If the earlier vehicle detected an obstacle or blockage along the road it dropped a marker by the side

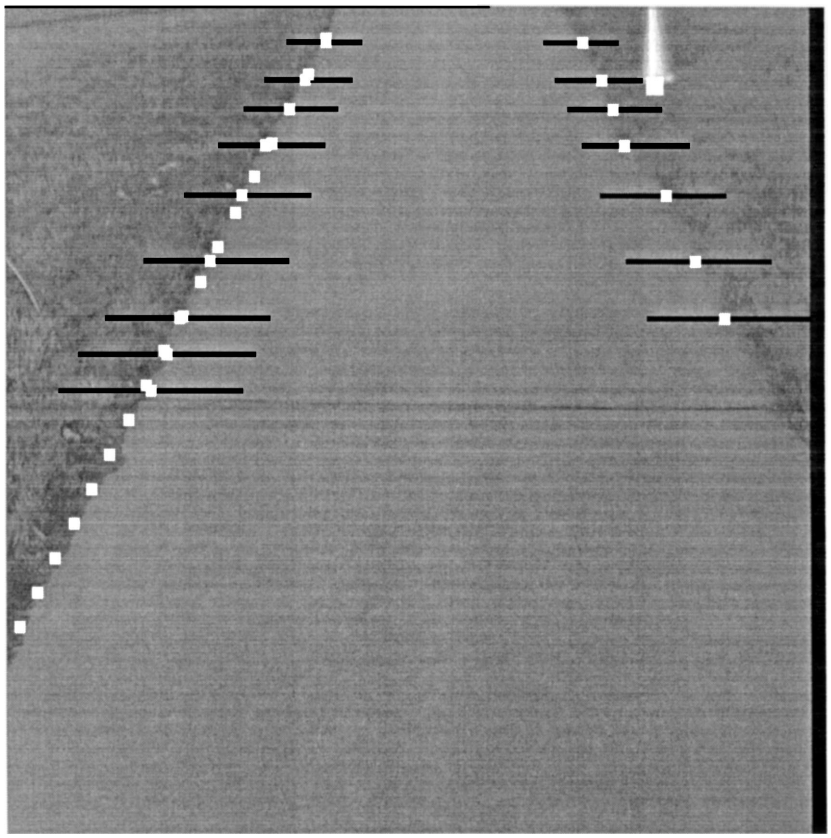


Figure 3. Road following and marker detection.

of the road. This marker served to warn the following robot that it should turn off the road and head cross-country towards a landmark in order to rendezvous. This was implemented in UMPRS via several plans and their enabling context conditions. The first plan implemented the case where the robot simply traveled all the way down the road, with the context condition that no marker left by the preceding vehicle had been spotted. The second plan implemented the case where the robot turned off the road, located the landmark, and drove the robot cross-country until it encountered the "other" vehicle, with the context condition that a marker left by the preceding vehicle had been spotted. Figure 3 shows a sample scene from one run. The white squares in the horizontal black bars show the sidewalk edge points found by YARF. Additional white squares mark YARF's least squares estimate of the location of the left edge of the sidewalk. The marker cone in the upper right of the image has been detected, and the center of its base marked with a white square.

In our experiments, our unmanned vehicle would start down the road, and could continue to the end of the road if it never saw the cone. When it encounters the cone, however, its UMPRS process sifts through the vehicle goals and the currently enabled plans for them to segue out of the road-scouting activity and into the rendezvous activity. These experiments thus showed the viability of using UMPRS for real-time reactive control of an unmanned vehicle. They also demonstrated the advantage of the procedural planning approach in providing for reactions to events in the environment that are not tied to prespecified locations, in contrast to map-based approaches like the Annotated Map formalism (Gowdy and Thorpe, 1990).

Plan Recognition for Coordinating Indoor Robots

To study the viability of using information assimilation techniques to support the coordinated execution of robotic plans, we implemented our belief-network-based plan recognition techniques as part of the sensing process of an indoor robot. The goal of this robot was to observe the movements of another robot (with a blatant

bar-code to be scanned), to infer the likely destination of that other robot, and to move to a complementary destination (typically, the same destination for a rendezvous). In our simulation experiments, we identified characteristics of the plan-recognition problem that impact decisions as to how long the observer should wait (how sure it should be) before acting upon its projection of where the observed agent is headed (Huber and Durfee, 1995). However, realizing these mechanisms in real indoor robots introduced further observational uncertainties that at times would undermine the inferences. Part of the information assimilation process, therefore, involved combining knowledge of relative vehicle movement abilities and models of sensor uncertainty, together with actual observations, to reach better hypotheses about vehicle movements.

Our experiments showed that the robots could indeed use knowledge about possible destinations along with the information assimilation techniques described above to efficiently rendezvous without explicit communication. Thus, the efficacy of information assimilation in support of planning in domains where explicit communication might be dangerous or impossible was established.

Integrated Multivehicle Mission Planning and Coordinated Execution in ModSAF

A true evaluation of our technologies in a military setting would involve four (working) UGVs with reasonable sensing capabilities operating in an environment where some unexpected contingencies would arise—a tall order. To approximate such a situation, we adopted the use of the ModSAF simulation system (Calder et al., 1993) as a powerful, widely used simulator for military purposes.

The ModSAF simulator provides a rich environment to test and evaluate our system. The simulator allows for one to create vehicles from over 200 different platforms ranging from aircraft to land-based vehicles. For our system, we chose to use UGV vehicles. These UGV vehicles were not endowed with all of the sensors and equipment that the real UGV vehicles contained, but they did have enough sensors to perform the kinds of tasks we were interested in. In ModSAF, vehicles are controlled by Finite State Machines, where the user gives the vehicle a sequence of tasks and the vehicle blindly executes them. For our purposes, we needed to have more control over the vehicle so we built into the simulator the ability for the vehicle to received commands from the UMPRS system, along with the ability to run some FSM tasks such as moving to specified points. In that way, we did not have to reimplement methods for directly controlling the vehicles' navigation: UMPRS could employ these methods when sending low-level commands such as go to some point, turn on and off sensors, read sensors and position of the vehicle, and send communications to other vehicles.

The setup of our system is summarized in Fig. 4. Currently, we have four vehicle agents and one OWS agent, running UMPRS. The OWS agent connects to the legacy graphical user interface (GUI) from Hughes STX. The IADB has been integrated in, along with a route planner and an observation point planner (OPP). The plan recognition subsystem for information assimilation is being integrated, but the integration is not complete at the time of this writing. The knowledge possessed by the agents is fairly impoverished: the OWS is knowledgeable about only a few kinds of missions, and acquires information about those missions therefore in mundane ways (without any need to disambiguate among choices). The UGVs have knowledge to achieve basic mobility and perception goals.

We have tested our system in scenarios ranging from 1 to 4 UGVs. A typical scenario is as follows (Fig. 5). An operator wants to scout ahead across enemy lines, such that the scouts eventually take up observation posts overlooking a suspected enemy encampment. This goal is selected from among the (currently few) options known to the mission planner, and the planner retrieves an appropriate procedure for generating the mission through binding map measures to variables, through prompting the operator for more data, and through invoking other planning tools (such as deciding placements of vehicles during a bounding overwatch). The procedure is executed, and the mission plan is formed in terms of a series of subgoals associated with each vehicle. These subgoal sequences are sent to the different UGVs, which run their own UMPRS processes to elaborate these further into specific robotic behaviors (and, along with these, their associated preplanned contingency behaviors). These behaviors are downloaded to ModSAF, and the vehicles begin executing their plans.

In the course of moving to their ultimate destinations, vehicles eventually become aware of an unexpected enemy in a nearby location. The sensory uncertainty captured in ModSAF means that when, during the mission, this happens is non-deterministic. Upon discovering

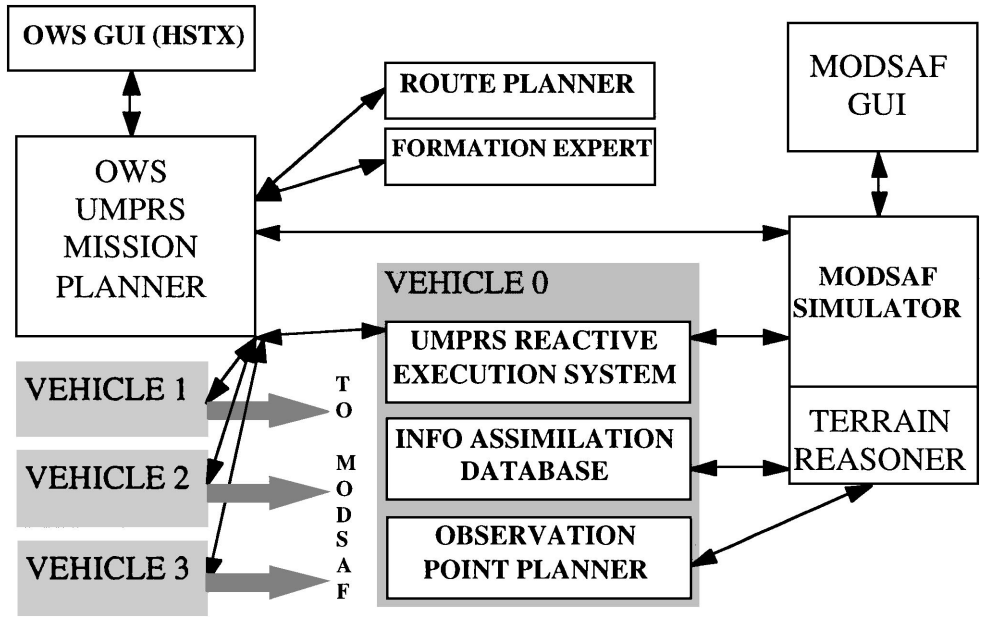


Figure 4. Demonstration setup.

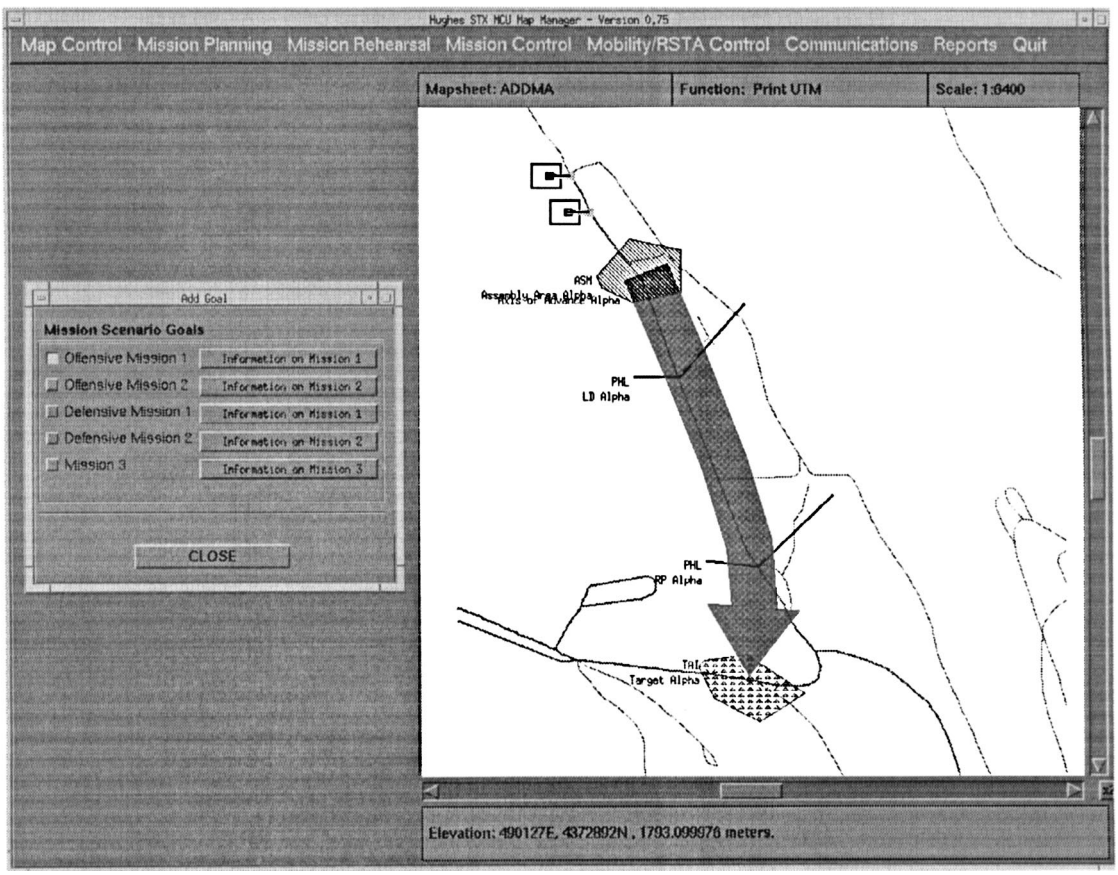


Figure 5. Mission planning display.

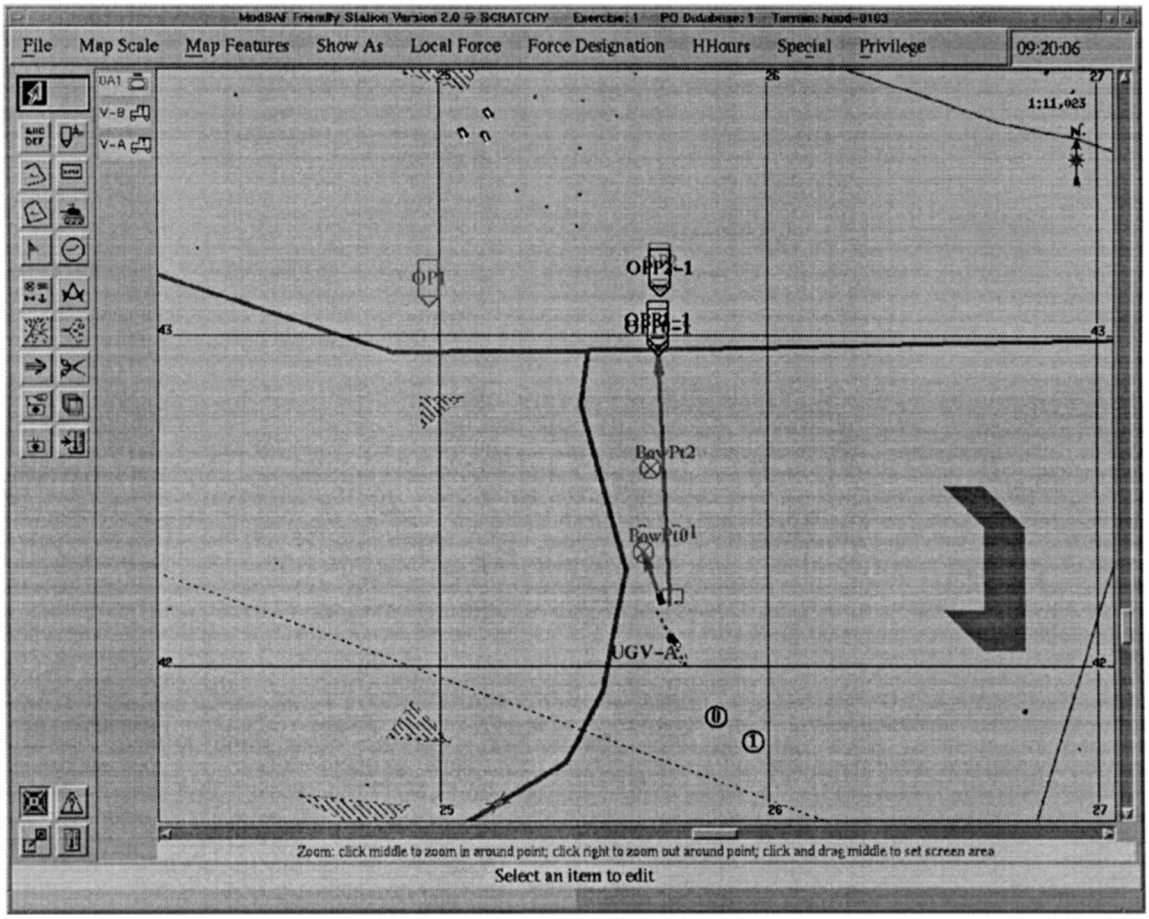


Figure 6. Mission execution in ModSAF.

the enemy, the vehicles alert each other and fall back to previous checkpoints.[1] This is a predefined contingency response, and so is executed very quickly. As they fall back, however, the updated IADB is probed by the UMPRS process on-board the vehicles, and UMPRS on the vehicle that first spotted the enemy feeds the new information into the OPP, which returns new observation points for the vehicle such that it can better isolate the extent of the unexpected enemy presence. The vehicle then leaves its fallback point and carries out this revised mission (Fig. 6).

We have demonstrated this kind of scenario for from one to four vehicles, illustrating the robust accomplishment of critical missions given our techniques. Variations on this theme can have the feedback go up to the IADB at the OWS, and the OWS could plan new observation points for several vehicles rather than having a single vehicle take on the task alone. Or the feedback can go up such that the vehicles wait at their fallback points until the operator explicitly downloads new robotic plans.

The strength of our approach can be appreciated by comparing it to the mission planning as done with the real UGVs during a demonstration led by Lockheed Martin (LM). In the LM demonstration, the development of a mission plan for the three vehicles could take several hours of two operators' time. The operators would micromanage the vehicles during execution. Sometimes they would get calls from the field, requesting that a vehicle be inched forward a little to look over a ridge. Sometimes they would use views from the UGV camera to realize that the UGV was going to the wrong place, and would quickly generate and download a new sequence of robotic actions.

In contrast, by incorporating knowledge into the OWS and UGVs, the demands on operator attention

during mission generation and execution are greatly reduced. In cases where the mission template is already known to the OWS, it is only a matter of minutes. And, during execution, more responsiveness is embedded in the automation tools, allowing much more rapid (and less operator-intensive) retasking.

The experiments from the fielded prototype indicated a strong interest on the part of the military personnel for these systems. Even despite the sometimes slow deployment of the UGVs and the intense monitoring costs on the operators, the use of the technology was viewed as a major win because it reduced casualties in a wargame scenario. A major complaint was about the effort required to formulate and monitor UGV missions, however, and it is for this reason that we expect the agent-based technologies that we have developed to have a significant impact on future incarnations of the UGV system. Our demonstrations to military personnel have elicited similar predictions from them as well.

Lessons Learned

To some extent, the lessons learned, as outlined in this paper, touch on social, rather than technical, issues of satisfying operator needs and embracing legacy systems. However, the technical accomplishments of this project provide a flexible means of satisfying these needs, as well as providing the foundation for integrating and coordinating the capabilities of robots, humans, and information systems.

The first lesson that we have learned is that knowledge-based plan execution systems, such as UMPRS, can provide a powerful, tailorable substrate for operator interaction. Systems such as PRS and Soar have already demonstrated ability to control systems in problems including malfunction diagnosis and handling (Ingrand et al., 1992), aircraft sequencing (Rao and Georgeff, 1995) and air combat (Rao et al., 1992). At the level of multivehicle control, our approach is similar to others; we have also demonstrated, however, that the reactive plan execution capabilities can be concurrently employed for operator interaction to provide mixed-initiative definitions of plans and constraints. These capabilities have found other uses beyond UGVs (Durfee et al., 1997).

A related lesson is on the importance of acquiring and propagating intent. Because vehicles might only intermittently be in touch with each other, it is critical that each know enough about the collective goals to forge ahead alone. The robust accomplishment of missions despite communication failures is in no small way due to our strategy of propagating goals and constraints, rather than commands, through the hierarchy.

Of course, properly coordinating the reactive execution of plans requires that the vehicles have sufficient understanding of the evolving plans and goals of others. Plan recognition is thus a critical part of the process of assimilating sensory information. Though we have not gone into the details of the process, our system provides agents with the ability to transform a library of executable plans into a belief-network representation that is especially well-suited to plan recognition (Huber et al., 1994).

Finally, having recognized (or received through communication) the plans that others are pursuing, coordinated execution requires an analysis of the possible outcomes of joint actions, and the establishment of commitments to prevent conflicts. Our experience has shown that there is no single best way to accomplish this in a complex, time-critical domain. Offline commitment to contingency plans, and to constraints of the degree to which plan deviations will be tolerated is one important mechanism. Centralized runtime replanning is another. A hybrid method that we have developed for this project has been to allow runtime examination of agents' adopted reactive plans to establish minimally limiting short-term constraints on reactions (Lee, 1996).

Conclusions

An agent-based approach to bridging the gap between operators and robotic vehicles provides significant advantages by embedding knowledge and initiative within a semi-automated mission planning and execution system. The future objectives of this work include deploying this system for controlling real UGVs in real military settings, and using both such a deployed system and the current simulation system that we have to evaluate the military benefits of using UGVs. Moreover, these technologies apply to the decentralized and semi-autonomous control of many other kinds of systems, and ideas from this work have been applied in other domains including automated ship systems (Durfee et al., 1997).

At a more fundamental level, an open question is the degree to which authority and responsibility should be shared between the operator, the operator's interface agent process (at the OWS), and the agent processes onboard the vehicles. In part, the answer to this question

is sociological. However, there are technological factors in modeling the impact of different divisions of autonomy on mission accomplishment time and success rate, and in the degree to which a better representation of the reactive plans (Lee and Durfee, 1994) can allow the division of autonomy to be divided dynamically, that provide exciting opportunities for future research.

Acknowledgments

This is a revised version of a paper presented at the ACM Autonomous Agents '97 Conference. We would like to acknowledge the contributions to this effort made by numerous people who have been part of our team over the years, especially Marc Huber and Jaeho Lee. This work has been supported, in part, by DARPA under contract DAAE07-92-CR012.

Note

1. When plan recognition is in place, a simple test will be that a vehicle that sees the enemy will fall back, and other vehicles will observe that it is moving to an unexpected destination and surmise (using the belief network) that this is strong evidence that it has seen an enemy.

References

Balch, T. and Arkin, R.C. 1995. Motor schema-based formation control for multiagent robot teams. In *Proc. of the First Int. Conf. on Multi-Agent Systems*, pp. 10–16.

Calder, R.B., Smith, J.E., Courtemanche, A.J., Mar, J.M.F., and Ceranowicz, A.Z. 1993. ModSAF behavior simulation and control. In *Proc. of the Third Conference on Computer-Generated Forces and Behavioral Representation*, Orlando, FL, pp. 347–356.

Cook, D., Gmytrasiewicz, P.J., and Holder, L. 1996. Decision-theoretic cooperative sensor planning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):1013–1023.

Durfee, E.H., Huber, M.J., Kurnow, M., and Lee, J. 1997. TAIPE: Tactical assistants for interaction planning and execution. In *Proc. of Autonomous Agents '97*.

Georgeff, M.P. and Lansky, A.L. 1986. Procedural knowledge. In *Proc. of the IEEE*, 74(10):1383–1398.

Gowdy, J. and Thorpe, C. 1990. Annotated maps for autonomous land vehicles. In *Proc. of the 1990 IEEE Conference on Systems, Man, and Cybernetics*, pp. 282–288.

Huber, M.J. 1996. Plan-based plan recognition models for the effective coordination of agents through observation. Ph.D. Dissertation, Univ. of Michigan.

Huber, M.J., Durfee, E.H., and Wellman, M.P. 1994. The automated mapping of plans for plan recognition. In *Proc. of the 1994 Conf. on Uncertainty in Artificial Intelligence*, pp. 344–351.

Huber, M.J. and Durfee, E.H. 1995. Deciding when to commit to action during observation-based coordination. In *Proc. of the First Int. Conf. on Multi-Agent Systems*, pp. 163–170.

Ingrand, F.F., Georgeff, M.P., and Rao, A.S. 1992. An architecture for real-time reasoning and system control. *IEEE Expert*, 7(6):34–44.

Kluge, K., Weymouth, T., and Smith, R. 1994. Information assimilation research at the University of Michigan for the ARPA unmanned ground vehicle project. In *Proc. of SPIE Sensor Fusion VII*.

Kluge, K. and Thorpe, C. 1995. The YARF system for vision-based road following. *Mathematical and Computer Modeling*, 22(4–7):213–233.

Lee, J. 1996. An explicit semantics for coordinated multiagent plan execution. Ph.D. Dissertation, Univ. of Michigan.

Lee, J., Huber, M.J., Durfee, E.H., and Kenny, P.G. 1994. UM-PRS: An implementation of the procedural reasoning system for multirobot applications. In *AIAA/NASA Conf. on Intelligent Robots in Field, Factory, Service, and Space*.

Lee, J. and Durfee, E.H. 1994. Structured circuit semantics for reactive plan execution systems. In *Proc. of the 1994 National Conf. on Artificial Intelligence (AAAI-94)*, pp. 1232–1237.

Rao, A.S., Moreley, M., Selvestrel, M., and Murray, G. 1992. Representation, selection, and execution of team tactics in air combat modelling. In *Proc. of the Fifth Australian Joint Conf. on Artificial Intelligence*, pp. 185–190.

Rao, A.S. and Georgeff, M.P. 1995. BDI agents: From theory to practice. In *Proc. of the First Int. Conf. on MultiAgent Systems*, pp. 312–319.

Shoham, Y. and Tennenholtz, M. 1994. On social laws for artificial agent societies: Off-line design. *Artificial Intelligence*, 72(1–2):231–252.

Stentz, A. 1990. The CODGER system for mobile robot navigation. *Vision and Navigation: The Carnegie Mellon Navlab*, C.E. Thorpe (Ed.), Kluwer Academic Publishers.

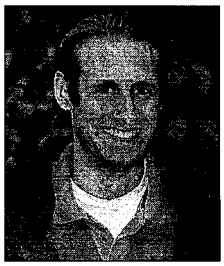
Stentz, A. 1995. Optimal and efficient path planning for unknown and dynamic environments. *Int. Journal of Robotics and Automation*, 10(3):89–100.

Tambe, M. 1995. Recursive agent and agent-group tracking in a real-time, dynamic environment. In *Proc. of the First Int. Conf. on MultiAgent Systems*, pp. 368–375.

Tambe, M., Johnson, W.L., Jones, R.M., Koss, F., Laird, J.E., Rosenbloo, P.S., and Schwamb, K. 1995. Intelligent agents for interactive simulation environments, *AI Magazine*, 16(1):15–39.

Edmund H. Durfee is an Associate Professor of Electrical Engineering and Computer Science and holds a joint appointment at the School of Information at the University of Michigan. He received his A.B. degree in Chemistry and Physics from Harvard University in 1980, and his M.S. and Ph.D. degrees from the University of Massachusetts in Computer Science and Engineering in 1984 and 1987, respectively. His research interests are in distributed artificial intelligence, planning, cooperative robotics, and real-time problem solving. He is a 1991 recipient of a Presidential Young Investigator award from the National Science Foundation to support this work,

and was an invited speaker at the National Conference on Artificial Intelligence in 1992. He is also an associate editor for IEEE Transactions on Systems, Man, and Cybernetics, and is the program co-chair for the 1998 International Conference on Multi-Agent Systems.



Patrick G. Kenny is a staff researcher at the University of Michigan AI Lab. He received his Bachelor Degree from the University of Minnesota in Computer Science. His interests are in Autonomous Agent Architectures, Intelligent control of Robots and Distributed Minirobotics.

Karl Kluge received his BS degree in Computer Science from Michigan State University in 1985, and his MS degree in Computer Science from Carnegie Mellon in 1988. He received his Ph.D. in Computer Science from Carnegie Mellon in 1993 for his work on the YARF vision-based road following system. He is currently an assistant research scientist in the Electrical Engineering and Computer Science Department at the University of Michigan, although he still roots for Michigan State. His research interests are computer vision, robot navigation, observation planning and sensor fusion, and planning/perception interactions. He is a member of the IEEE.