A. Innovative Claims

The College of Computing of the Georgia Institute of Technology is pleased to submit this proposal entitled *Multi-level Learning in Hybrid Deliberative/Reactive Mobile Robot Architectural Software Systems* to DARPA in response to BAA99-09. We propose to develop revolutionary learning methods in the context of proven DARPA-developed software to accomplish complex adaptive autonomous mobile robot operations in dynamic, hostile environments. We will develop powerful platform-independent multi-level learning strategies that are appropriate for battlefield operations. Using a carefully designed and validated mission planning and control framework, warfighters will rapidly task their supporting robot team members to perform dangerous duties, leveraging novel machine learning assets introduced into our existing *MissionLab* software system.

This effort will provide the following revolutionary learning and adaptation capabilities:

- Probabilistic situational recognition and indexing into behavioral sets for opportunistic planning and reaction.
- The use of case-based reasoning (CBR) methods for situation-dependent behavioral gain and assemblage switching at the reactive level of execution.
- The use of CBR for reasoning in Finite State Automata (FSA) plan generation through the use of "wizards" to guide high-level deliberative planning.
- Specialized reinforcement learning methods ("learning momentum") to adjust behavioral gains at run-time.
- The integration of Q-learning methods for behavioral assemblage selection at a level above gain adjustment.

Publications are available on the results already obtained for most of these methods when considered in isolation [17,19,30,36,45,46,48,67,68,69]. These learning techniques will be integrated within our existing *MissionLab* software system developed under previous and ongoing DARPA funding, which to date has not addressed machine learning issues.

Our research group at Georgia Tech has been working on hybrid deliberative/reactive robotic systems since 1987. The proposed research leverages our extensive experience gained from fielding a range of successful systems, including: autonomous formation control for two HMMWVs that was demonstrated live to a military audience during the UGV Demo II program; winning multi-robot teams at the AAAI-94 and AAAI-97 mobile robot competitions; and numerous laboratory demonstrations using our 3 Denning and 5 Nomad robots to display results such as team teleautonomy, multiagent mission specification, team communication minimization, formation control, etc. We are also participants in the ongoing DARPA Tactical Mobile Robotics Program providing real-time reactive behaviors, mission specification capabilities, communication minimization and planning abilities, and real-time analysis tools. Much of the hardware and software in place from these two prior programs and others can also be utilized for this research, providing a quick, efficient and cost-effective solution to the needs of the DARPA MARS program.

B. Technical Rationale and Approach

B.1 Preliminary Work

We first review our earlier robot software research funded by DARPA's Real-time Planning and Control Program in support of the Unmanned Ground Vehicle (UGV) Demo II program, and DARPA's Tactical Mobile Robotics Program.

B.1.1 Mobile Autonomous Robot Software Systems

A pressing problem for the Department of Defense in particular and for robotics in general is how to provide an easy-to-use method for programming robots, making these systems more accessible to the soldier. Toward that end, the *MissionLab* mission specification system has been developed [57]. An agent-oriented philosophy is used as the underlying methodology, permitting the recursive formulation of entire societies of robots.

A society is viewed as an agent consisting of a collection of either homogeneous or heterogeneous robots. Each individual robotic agent consists of assemblages of behaviors, coordinated in various ways. Temporal sequencing [13] affords transitions between various behavioral states which are naturally represented as a finite state acceptor. Coordination of parallel behaviors can be accomplished via fusion, action-selection, priority, or other means as necessary. These individual behavioral assemblages consist of groups of primitive perceptual and motor behaviors which ultimately are grounded in the physical sensors and actuators of a robot.

An important feature of *MissionLab* is its ability to delay binding to a particular behavioral architecture (e.g., schema-based, MRPL (used in Demo II)) until after the desired mission behavior has been specified. Binding to a particular physical robot also occurs after specification, permitting the design to be both architecture- and robot-independent.

MissionLab's architecture appears on the left of Figure 1. Separate software libraries exist for abstract behaviors, and the specific architectures and robots. The user interacts through a design interface tool (the configuration editor) which permits the visualization of a specification as it is created. The right side of Figure 1 illustrates an example *MissionLab* configuration that embodies the behavioral control system for one of the robots capable of conducting explosive ordnance disposal (EOD) operations that was employed for testing in usability studies [56]. The individual icons correspond to behavior specifications which can be created as needed or preferably reused from an existing repertoire available in the behavioral library. Multiple levels of abstraction are available, which can be targeted to the abilities of the designer, ranging from whole robot teams, down to the configuration description language for a particular behavior, with the higher levels being those easiest to use by the average soldier.

After the behavioral configuration is specified, the architecture and robot types are selected and compilation occurs, generating the robot executables. These can be run within the simulation environment provided by *MissionLab* (Fig. 2 left) or, through a software switch, they can be downloaded to the actual robots for execution (Fig. 2 right).

At the reactive run-time level when using motor-schema based control [5], primitive behaviors are specified for a robot which yield global societal task-achieving action in an unstructured environment. Reactive behavioral control is now a well established technique for providing rapid real-time response for a robot by closely tying perception to action. Behaviors, in various forms, are the primary building blocks for these systems. Schema-based systems are a form of reactive behavioral control that are further characterized by their neu-

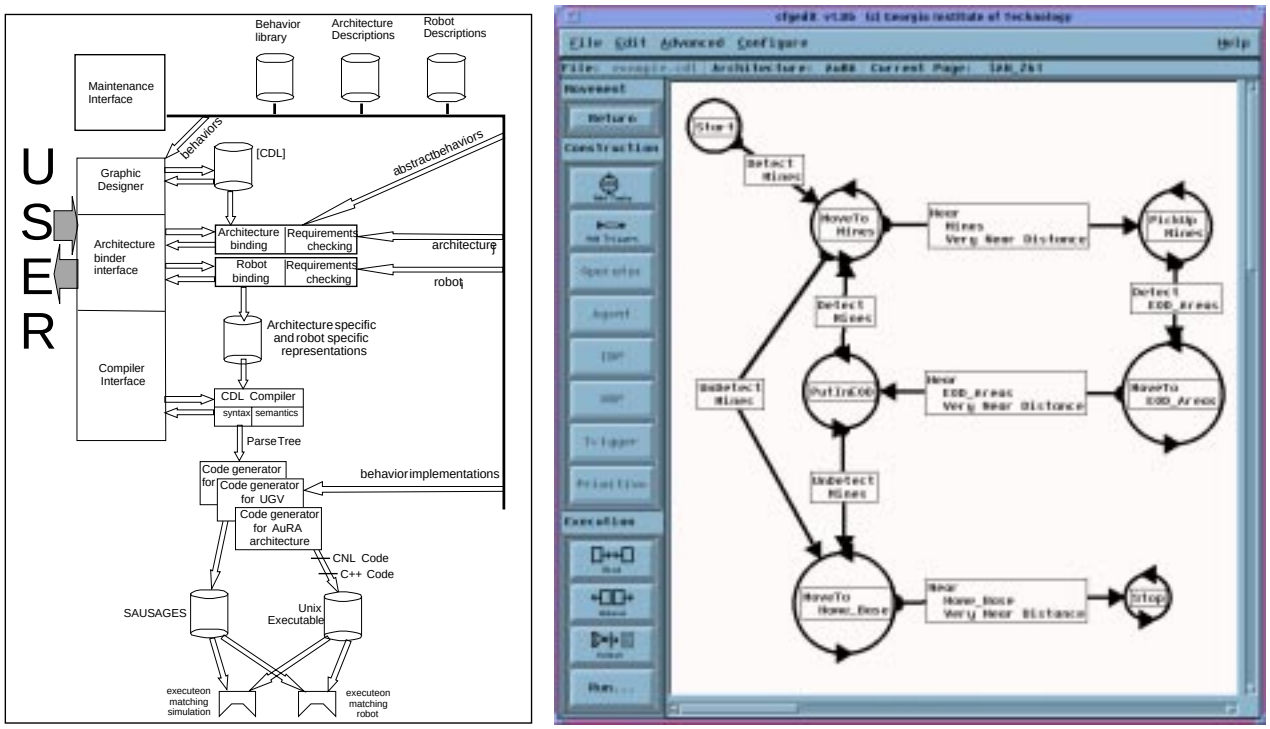

Figure 1: *MissionLab*. (See text for description).

roscientific and psychological plausibility, the absence of arbitration between behaviors, the fusion of behavioral outputs through the use of vector summation analogous to the potential fields method, inherent flexibility due to the dynamic instantiation and deinstantiation of behaviors on an as-needed basis, and easy reconfigurability through the use of high-level planners or adaptive learning systems.

MissionLab was demonstrated at UGV Demo C in the Summer of 1995 to military personnel and again at the concluding Demo II workshop in June 1996 and is continuing active development under DARPA's Tactical Mobile Robot Program. *MissionLab* is available via the world-wide web at: http://www.cc.gatech.edu/aimosaic/robot-lab/research/MissionLab.html

The current version software environment being used within DARPA's Tactical Mobile Robotics Program for our research, contains 3 major subsystems: Executive, Premission, and Run-time (Fig. 3). The Executive subsystem is the major focus for operator interaction, providing an interface to both the run-time simulators and actual robot controllers as well as the premission specification facilities. The Premission subsystem's role is to provide an easy-to-use interface for designed robot missions and a means for evaluating its usability. The Run-time control system, which is located on each active robot, provides the execution framework for enacting reactive behaviors, acquiring sensor data and reporting back to the Executive subsystem to provide situational awareness to the team commander. Additionally, a separate support system is provided for interprocess communications. Details on current system specifications are available at http://www.cc.gatech.edu/ai/robot-lab/tmr.

Figure 2: Left: Simulated Run on Denning Robot. Right: The same code executed on actual Denning Robot

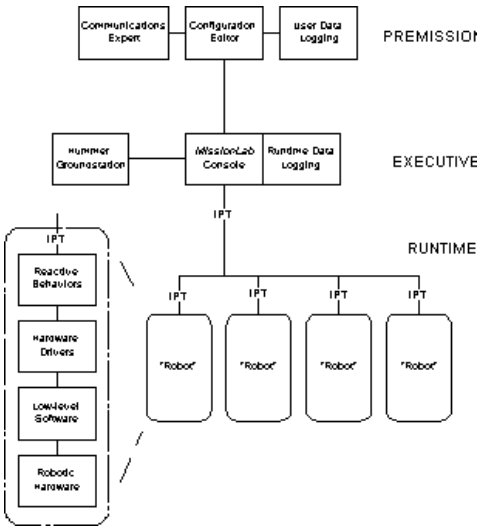

Figure 3: Georgia Tech Software Specification for DARPA TMR Program

B.1.2 Hybrid Deliberative/Reactive Architectures

Our research on robotic architectures pioneered the notion of deliberative/reactive systems through the design of the Autonomous Robot Architecture (AuRA), which *MissionLab* embodies. AuRA was developed in the mid-1980's as a hybrid approach to robotic navigation [3,4]. Incorporating a conventional planner that could reason over a flexible and modular behavior-based control system, specific robotic configurations could be constructed that integrated behavioral, perceptual and *a priori* environmental knowledge [9]. Initially, hybridization arose from the presence of two distinct components: a deliberative or hierarchical planner, based on traditional artificial intelligence techniques; and a reactive controller, based upon schema theory [2]. It was the first robot navigational system to be presented in this integrative manner [7]. This hybridization provides the entry point for the multi-level learning mechanisms put forward in this proposal.

Schematically, the components of AuRA are depicted in Figure 4. Two major planning and execution components are present: a hierarchical system consisting of a mission planner, spatial reasoner, and plan sequencer, coupled to a reactive system, the schema controller. In

the style of a traditional hierarchical planning system [1,60,73], the highest level of AuRA is a Mission Planner concerned with establishing high level goals for the robot and the constraints within which it must operate. In AuRA-based systems constructed to date, the Mission Planner has acted primarily as an interface to a human commander. The Spatial Reasoner, originally referred to as the Navigator [4], uses cartographic knowledge stored in long-term memory to construct a sequence of navigational segments which the robot must execute in order to complete its mission. In the first implementation of AuRA, this was an A* planner operating over a meadow map (hybrid free space/vertex graph) representation [6]. The Plan Sequencer, referred to as the Pilot in earlier work, translates each path leg generated by the Spatial Reasoner into a set of motor behaviors for execution. In the original implementation, the Plan Sequencer was a rudimentary rule-based system. More recently it has been implemented as a finite state sequencer [57]. Finally, the collection of behaviors (schemas), specified and instantiated by the Plan Sequencer, is then sent to the robot for execution. At this point, deliberation ceases, and reactive execution begins.

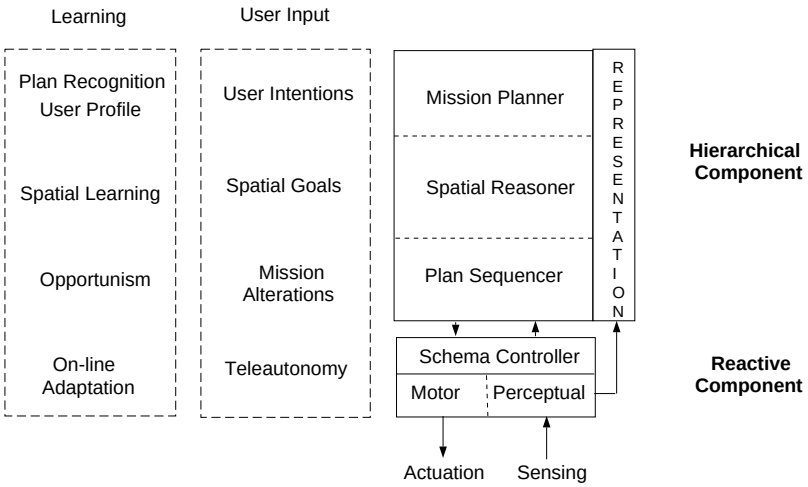

Figure 4: High-level AuRA Schematic

The schema manager is responsible for controlling and monitoring the behavioral processes at run-time. Each motor behavior (or schema) may be associated with one or more perceptual schemas capable of providing the sensory stimulus required for that particular behavior. This action-oriented perception is the basis for this form of behavior-based navigation [8]. Each behavior generates a response vector in a manner analogous to the potential fields method. The schemas operate asynchronously, transmitting their results to a process (**move-robot**) which sums and normalizes these inputs and transmits them to the low-level control system for execution. Of late, additional behavioral coordination mechanisms have been introduced including priority-based arbitration and action-selection.

AuRA is highly modular by design. Components of the architecture can be replaced with others in a straightforward manner. This is particularly useful in research. Various deliberative planners including multistrategy planners and plan sequencers have readily alternated with the original version over the years. Some examples include:

- A specialized Mission Planner developed for an assembly task where boxes are pushed together into a specified arrangement. This planner was ported to a Denning mobile robot that competed in the 1993 AAAI Mobile Robot Contest. The planner was further extended in [77] to learn and reason over more general planning tasks.

- The original A* Spatial Reasoner has been replaced with Router [36], a multi-strategy planner. Router models navigable routes as topological links between nodes instead of the metric meadow-map representation used previously. The system was tested on a Denning mobile robot which successfully navigated from room to room and down corridors in an office/laboratory building.
- Perceptual schemas have been expanded to incorporate specialized action-oriented sensor fusion methods [63]. Recognizing that in many cases multiple sensors sources are better than individual ones, specialized strategies were developed to fuse data together within the context of action-oriented perception. Dempster-Shafer statistical methods provided the basis for evidential reasoning [64].
- The original rule-based Plan Sequencer has been replaced with a temporal sequencer [13] based on finite state acceptors (FSAs) [57]. The FSA is an expression of a plan, where each state represents a specific combination of behaviors that accomplish one step of the task. Transitions are made from one state to another when significant perceptual events trigger them.

Another strength is the flexibility AuRA provides for introducing adaptation and learning methods, which situates it perfectly for the research embodied in this proposal. In early implementations of AuRA, learning arose only from short-term memory of spatial information [34] used for dynamic replanning. Since then, a variety of learning techniques have been introduced that serve as the springboard for the research in this proposal, including:

- On-line adaptation of motor behaviors using a rule-based methodology [30].
- Case-based reasoning methods to provide discontinuous switching of behaviors based upon the recognition of new situations [67].
- Genetic algorithms that configure the initial control system parameters in an efficient manner [68] and that allow a robot to evolve towards its ecological niche in a given task-environment.

These, however, have not been well integrated into the complete architecture to date, but nonetheless serve as proofs of concept for much of the research and development proposed herein.

The generalizability of AuRA to a wide range of problems is another strength. Various architectural components have been applied in a variety of domains including: manufacturing environments [15,14]; three dimensional navigation as found in aerial or undersea domains [10]; indoor and outdoor navigation [4]; robot competitions [12,22]; vacuuming [58]; military scenarios [57,21]; mobile manipulation [27]; and multi-robot teams [11,20].

In summary, the major strength of the Autonomous Robot Architecture results from the power of wedding two distinct AI paradigms: deliberation and reactivity. AuRA provides a framework for the conduct of a wide range of robotic research including deliberative planning, reactive control, homeostasis, action-oriented perception, and machine learning. AuRA has been motivated but not constrained by biological studies, drawing insight wherever available as a guideline for system design. Other strengths lie in AuRA's modularity, permitting ready integration of new approaches to various architectural components; flexibility as evidenced by the ease of introduction of various learning methodologies and novel behaviors; and generalizability as demonstrated by its applicability of a wide range of domains including robot competitions among others.

B.1.3 Adaptation and Learning in Robotic Systems

Our research program has investigated two fundamental types of learning mechanisms in intelligent robotics systems: (i) on-line adaptive learning, which allows a system to respond to unexpected situations and learn while engaged in the problem-solving process, and (ii) off-line learning, which allows a system to learn through slower, more intensive learning processes. Learning is carried out using multiple learning methods, including case-based reasoning, reinforcement learning, and genetic algorithms. Of particular concern is the development of methods that allow a reactive control system to improve its performance by learning to tune its behavior to different environments. This requires the system to learn a model of not just the environment but also the interaction between the system and the environment in a continuous, on-line manner. However, to ensure that the system does not get bogged down in extensive high-level reasoning, the knowledge represented in the model must be based on perceptual and motor information easily available at the reactive level.

The requirements of continuous problem domains, such as robotic control, are significantly different in ways that do not permit ready application of traditional symbolic machine learning methods. For example, consider the problem of driving a car on a highway. Car-driving experiences can vary from one another in infinitely many ways. The speed of a car might be 55 mph in one experience and 54 mph in another. Within a given episode, the speed of the car might continuously vary, both infinitesimally from moment to moment, and significantly from, say, the highway to an exit ramp. The problem solving and learning process must operate continuously; there is no time to stop and think, nor a logical point in the process at which to do so.

Such problem domains are "continuous" in three senses. First, they require *continuous representations*. For example, a robotic navigation task requires representations of perceptual and motor control information. The input is a continuous stream of perceptual data from ultrasonic and other sensors; the data itself is analog in the sense that the value of an input parameter can vary infinitesimally (within the limits of the digitization and sampling parameters). Second, continuous problem domains require *continuous performance*. For example, driving a car requires continuous, on-line action. Often, problem-solving performance is incremental of necessity because of limited knowledge available to the reasoning system and/or because of the unpredictability of the environment; the system can at best execute the "best" short-term actions available to it and then re-evaluate its progress. A robot, for example, may not know where obstacles lie until it actually encounters them. Third, these problem domains require *continuous adaptation and learning*. As the problems encountered become more varied and difficult, it becomes necessary to use fine-grained, detailed knowledge in an incremental manner to act, and to rely on continuous feedback from the environment to adapt actions and learn from experiences.

We have developed novel learning methods using techniques from case-based reasoning, reinforcement learning, and genetic algorithms. Implemented systems include PEPE, a personal robotic pet; ACBARR [67], a case-based reactive navigation system; SINS [69], a multistrategy case-based and reinforcement learning system that continuously constructs and adapts its own system-environment model; and GA-ROBOT [68], a genetic algorithm for evolving reactive control systems.

B.1.4 Usability Evaluation in Robot Programming Toolsets

To ensure that robot commanders who are not trained experts in robotics can use *MissionLab* and similar systems, formal usability testing [42,61] was performed. Two experiments were designed to determine: (1) if it was faster to specify robot missions using the *Mis-*

sionLab toolset than by writing corresponding C code; and (2) to measure the ability of non-programmers to specify robot missions using *MissionLab*.

The *MissionLab* experiments were carefully designed to minimize bias. They were conducted in Georgia Tech's Usability Laboratory which had one-way mirrors, sound and video recording equipment, and necessary computer and communications equipment. An independent third party observer conducted and monitored the experiments to ensure impartiality with all interaction between the observer and the participants carefully scripted to ensure consistency.

The first experiment conducted with *MissionLab* focused on determining the performance of novice and expert users creating solutions for five benchmark robot missions using the Configuration Editor. The second experiment measured the ability of these same participants to specify the same missions using `C++`. All solutions were evaluated using the *MissionLab* simulation system. The observer gave the participants one task description at a time and asked them to construct a configuration which achieved it. After completing as many of the tasks as possible within 2 hours, the session concluded with a survey.

Twelve people participated in the experiments, including one ROTC student, three already familiar with the *MissionLab* toolset, four with no programming experience (one of which was the ROTC student), and four programmers who had no previous *MissionLab* experience. These experiments showed that (1) it was faster to create missions using *MissionLab* than `C++` and that (2) non-programmers were able to specify simple robot missions with only 1/2 hour of training. A detailed development and analysis of the experiments can be found in [55,56].

B.2 Proposed Research and Technical Approach

Multi-level learning is proposed to occur within both the deliberative and reactive components of *MissionLab*'s embodiment of the hybrid philosophy of the AuRA architecture. Both are described below.

B.2.1 Reactive Learning Methods

Three basic types of learning are proposed in this research at the reactive level: (1) direct alteration of the parameters and gains of the constituent behaviors; (2) selection of appropriate behavior assemblages based on situational assessment; and (3) determination of a suitable sequence of behavioral assemblages given a particular task-environment.

1. Learning Momentum: This approach, based on gain alteration of individual behaviors, enables a reactive control system to adapt its behavior based on recent experience. When there are relatively few abrupt environmental discontinuities we have demonstrated that a reactive robot can adapt in a manner which leads to enhanced performance [30]. A key problem for these systems is not merely the selection of the appropriate behavior set but also determining values for behavioral characteristics that are well suited for a particular environment. Although in general these values are not overly sensitive and do not require extensive tuning to produce good results, it is still desirable to allow a reactive robotic system to respond to its environment by allowing these settings to be altered in response to both the robot's performance and changes in the world. Further, the robot may be required to navigate in unfamiliar environments where the appropriate values for the behaviors cannot be known in advance. Our research will concentrate on extending reactive controllers to include the ability to learn these values. The underlying strategy in our work is that if something is working well then continue doing it and try doing it a bit harder, and conversely, if

things are not proceeding well then try something different. In this way, a learning system builds *momentum* since success due to parametric adjustment leads to better performance. For behavior-based reactive systems, this readily translates into altering the schema gains and parameters continuously. Our system uses a rule-based methodology to alter the gain and parameter values incrementally based on current environmental conditions and past successes. Of special note is the system's ability to successfully navigate out of "box canyons" or dead end paths where a non-adaptive reactive controller would stall [30].

Of particular importance is the integration of these methods for adaptation into *MissionLab* in an easy-to-use fashion. Currently the results we have obtained have not been integrated into the systems we currently use for programming mobile robots in battlefield environments. To do so requires the development of an understanding of salient situations and their relationships to the behavioral performance of the robot. Using DARPA specified scenarios, we will identify situational characteristics, initially manually, followed by the use of automatic classification methods (e.g., AutoClass [28] or other techniques for situational characterization that have been pioneered in our laboratory [17]). These methods will then be tested on the testbed described in Section B.2.1.

2. CBR for Behavioral Selection:

Our first attempt at building an adaptive reactive navigation system focused on the issue of using case-based reasoning to guide reactive control. A central issue here is the nature of the guidance: At what grain size should the reactive control module represent its behaviors, and what kind of "advice" should the case-based reasoning module provide to the reactive control module? To investigate these issues, we built the ACBARR (A Case-Based Reactive Robotic) system which focused solely on using case-based reasoning to guide reactive control but did not deal with the issue of learning the cases necessary to do so.

In order to achieve more robust robotic control, ACBARR used sets of behaviors, called *behavior assemblages*, to represent appropriate collections of cooperating behaviors for complex environments; *behavior switching* to dynamically select behaviors appropriate for a given environment; and *behavior adaptation* to adapt and fine-tune existing behaviors dynamically in novel environments [67]. There are two types of behavior adaptations to be considered. One option is to have the system modify its current behaviors based on immediate past experience. This is a local response to the problem. A more global solution is to have the system select completely new assemblages of behaviors based on the current environment in which it finds itself. A robust system should be able to learn about and adapt to its environment dynamically in both of these ways.

While ACBARR demonstrated the feasibility of on-line case-based reasoning in systems requiring continuous, real-time response, it relied on a fixed library of cases that were hand-coded into the system. The system could adapt to novel environments—an important source of flexibility—but it could not improve its own adaptation behavior through experience. Since the knowledge required for behavior adaptation is stored in cases, we turned our attention to the problem of learning cases through experience. We built SINS (Self-Improving Navigation System), which is similar to the ACBARR system but can learn and modify its own cases through experience [67]. The representation of the cases in SINS is different and is designed to support learning, but the underlying ideas behind the two systems are very similar.

A primary motivation behind SINS was to avoid relying on hand-coded, high-level domain knowledge. There are several disadvantages of relying on such knowledge. First, it is based on an a priori model of the interaction of the robot and its environment. Such a model is only an approximation to reality and thus cannot cover all relevant aspects for

successful mission performance, especially in novel circumstances not anticipated by the system designer. Second, such a model is based primarily on the designer's understanding of the interaction between the robot and the environment, and thus the quality of the model depends highly on the knowledge and skills of the system designer. Third, it is unclear how a system could extract the necessary high-level knowledge from low-level sensory input in real-time; this is one of the standard motivations for developing reactive systems. Fourth, a user-designed high-level representation does not provide a scientific explanation of how such knowledge comes to be learned in the first place.

Thus, the representations used by SINS to model its interaction with the environment are initially under-constrained and generic; they contain very little useful information for the navigation task. As the system interacts with the environment, the learning module gradually modifies the content of the representations until they become useful and provide reliable information for adapting the navigation system to the particular environment at hand.

The learning and navigation modules function in an integrated manner. The learning module is always trying to find a better model of system-environment interaction so that it can tune the navigation module to perform better. The navigation module provides feedback to the learning module so it can build a better interaction model. The system's behavior is then the result of an equilibrium point established by the learning module, which is trying to refine the model, and the environment which is complex and dynamic in nature. This equilibrium may shift and need to be re-established if the environment changes drastically; however, the model is generic enough at any point to be able to deal with a very wide range of environments.

The navigation module in SINS can be adapted to exhibit many different behaviors. SINS improves its performance by learning how and when to tune the navigation module. In this way, the system can use the appropriate behavior in each environmental configuration encountered. The learning module, therefore, must learn about and discriminate between different environments, and associate with each the appropriate adaptations to be performed on the motor schemas. This requires a representational scheme to model, not just the external environment, but also the interaction between the system and the environment. However, to ensure that the system does not get bogged down in extensive high-level reasoning, the knowledge represented in the model must be based on perceptual and motor information easily available at the reactive level.

While ACBARR and SINS were both successful and demonstrated in both simulation and on actual robots, they were standalone systems not integrated into the context of an overall functional robotic system. Our plan under this proposal is to integrate the techniques from these systems into *MissionLab*, so that the adaptation and learning algorithms are active, under the user's behest, as the system performs its tasks. Of particular interest are tasks other than simple navigation, which our earlier systems focused on. *MissionLab* is capable of being used for a range of military scenarios, and we plan to demonstrate that our techniques can be integrated to yield improved performance.

3. Q-learning for Behavioral Assemblage Selection:

An alternative to learning at the parametric level is to use reinforcement methods (i.e., Q-learning) to determine the correct behavioral sequencing of behaviors. Recent work in our laboratory [17,19] has studied this problem in regards to robot specialization in teams, focusing on tasks such as robot soccer, formation maintenance, and foraging. This approach also allows robots to grow into specialized ecological niches that will fit their task/environment well.

Q-learning is where a single Q utility function is learned to evaluate both actions and states of the robot (Fig. 5b) [80]. A study by Lin [53] comparing Q- and Adaptive heuristic critic (AHC) reinforcement learning methods for enhancing the survivability of an agent in a simulated dynamic world provides evidence that Q-learning is the superior method for reactive robotic applications. Indeed, Q-learning currently dominates behavior-based robotic reinforcement learning methods (See Section G).

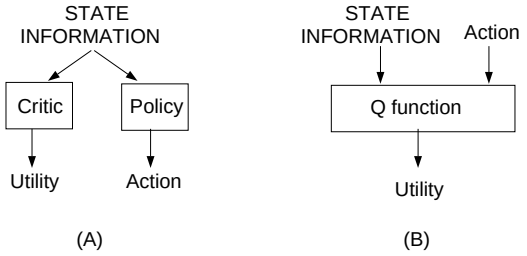

Figure 5: AHC (left) versus Q-learning (right)

Q-learning provides the ability to learn the correct global robotic response for a given set of stimuli presented by the world by determining which behavioral actions are most appropriate for a given situation. An update rule is used for the utility function $Q(x, a)$ where x represents the states and a the resulting actions:

$$Q(x, a) \leftarrow Q(x, a) + \beta(r + \lambda E(y) - Q(x, a)) \tag{1}$$

where: β is a learning rate parameter; r is the payoff (reward or punishment); λ is a parameter, called the discount factor, ranging between 0 and 1; $E(y)$ is the utility of the state y which results from the action and is computed by $E(y) = max(Q(y, a))$ for all actions a. Reward actions are also propagated across states so that rewards from similar states can facilitate their learning as well.

Research in our laboratory has already pioneered the application of Q-learning for learning behavioral diversity (e.g., specialized roles in robotic teams) [17,19]. We intend to refine and apply these methods to an integrated behavioral assemblage learning system within the confines of *MissionLab*. This will exploit the metrics and evaluation methods discussed in Sec. B.2.2-3 while exploiting the learning mechanisms already in place in our publically available software *Javabots*.

One of the major problems with Q-learning applications in robotics is the proliferation of input/output states. This is manageable in our system as it only needs to learn the correct sets of behaviors (assemblages) to use at a given time as opposed to very low-level reactions. Hence output discretization is not an issue, making reinforcement learning both feasible and rapid. One major focus of our work will be in identifying the high level environmental features that characterize the situation, not unlike the problem we are confronted with in the learning momentum task discussed earlier. Here, however, learning is not parametric. Instead, appropriate constellations of behaviors are learned to fit a task at the right time, and the focus on learning is the modification of the underlying perceptual triggers that cause these transitions between the various behavioral states [13]. Thus, this form of Q-learning operates at a level well above our work in learning momentum: not altering the behaviors themselves, but rather the selection mechanisms by which they are invoked, so multi-level mechanisms are at play even within the context of reactive learning, not only across the boundaries of deliberation and reactivity.

B.2.2 Deliberative Learning Methods

Two deliberative learning methods provide high-level plan generation and specification abilities through (1) the use of case-based reasoning, and (2) probabilistic methods for situation assessment.

1. CBR "Wizardry":
Case-based reasoning methods, used earlier in reactive learning, can also provide a means to guide plan specification. Currently within *MissionLab*, the end-user determines the plan through a graphical user interface [57]. While this is a relatively easy way to create a plan [56], it is not planning in and of itself. We propose that interactive CBR methods be developed that will generate the plan (encoded as an FSA), under end-user guidance. This interactive method will be along the lines of wizards commonly found in commercial software systems, with extensive help facilities. This makes the task of plan generation semi-automatic and very high-level, as opposed to the current manual specification methods now in place. This interface will also be subjected to usability testing as we have done for our previous interfaces to ensure end-user acceptance. It inherits all of the benefits of *MissionLab* for simulation and deployment of the resulting mission and it will become an integral component in a subsequent release.

What is required for the successful application of CBR to high-level mission selection is an effective representation of mission cases and an automated means for their capturing. Coupled with system integration and user interface design, these issues of knowledge representation and acquisition will form a primary aspect of this research. This must needs be focused on cases developed from DARPA requirements but the underlying methodologies we will develop and employ will transcend the relatively few missions to be used as exemplars within the MARS program itself. These CBR tools will be fully automated and tightly integrated within the *MissionLab* software framework.

2. Probabilistic Planning and Execution: Our current methods of behavioral selection are binary in nature: particular perceptual events and situations either induce a change in the plan (configuration) or they do not. It is our intention to use a "softer, kinder" means for matching situations and perceptual triggers to these transitions. Using techniques drawn from within our research group (e.g., hidden Markov models, Bayesian, and Dempster-Shafer statistical methods) we will devise effective means for learning and encoding probabilistic relationships of environmental features which comprise situations that can then be used to both plan and react given highly uncertain situational identification capabilities.

One of the advantages of schema-based navigation is that it works well in unstructured and only partially known environments. For example, if the robot encounters an obstacle, it simply activates an appropriate schema that avoids it. This property is important in environments that change quickly and can contain a large number of unexpected obstacles, such as war zones. In such worlds, it is impossible to predict exactly during planing how many obstacles there are and their distribution. It is therefore tempting to ignore them entirely during planning. For example, if the plan assumes that a particular part of the environment is relatively free of obstacles, it expects that it can traverse it quickly and might choose this route to achieve its mission. However, if that part of the environment is really cluttered with obstacles, then the traversal of the selected route is more time-consuming than expected. Consequently, if the planner had chosen a different route, the robot might have been able to complete its mission faster - and speed is often critical for military missions. Similarly, it is important to take into account during planning that a route might have to be abandoned

because it turns out to be untraversable during execution.

Our research addresses this problem by learning expectations about different areas of the environment, such as their obstacle densities or distributions and associated probabilities regarding traversability, and using them during planning. We will categorize areas of the environment into different classes and gather statistics for each class.

Based on our previous experience with probabilistic approaches to robot planning, navigation, and learning [45,46,47] (although in a framework different from schema-based navigation), we intend to provide three components integrable with *MissionLab*: software that learns characteristics of different categories, which assemblage of schemas to use for which category, and how well this assemblage performs; a plan-refinement module that uses this information to determine plans whose expected execution time is smaller than that of the plans that our current system finds; and a categorization module for the Executive subsystem that continuously determines the category of the current operating environment of the robot and chooses the most appropriate behavioral assemblages for the task-environment. In the following, we explain the three components in more detail:

1. We will provide integrated software that performs learning within the simulation environment that *MissionLab* provides or through the experience of actual robots that navigate several prototypical environments while gathering statistical data. These statistics include which assemblages performs best for each environmental category and which category is likely to follow a given category (for example, which kinds of obstacles often co-occur). In practice, it is important to be able to perform learning in the simulation environment because one might not have access to the actual operating environment of the robots prior to the mission, especially if it is a war zone, and might not have the time to create a realistic substitute. These simulation methods could be extended in out year options to include high-fidelity warfighting simulations such as JCATS.
2. We will provide software that performs probabilistic planning using the gathered statistics to find plans with a small expected execution time, by approximating the planning problem as a Markov Decision Process (MDP) model which is then solved using MDP methods. Our planner will be able to reason with these probabilities. For example, it will be able to take into account the possibility that a planned route might have to be abandoned because it turns out to be untraversable.
3. We will provide navigational software using Bayesian techniques that are able to quickly classify the current operating environment of the robot, which enables it to choose the best behavioral assemblages according to a given mission scenario.

B.2 Experimental Testbed and Evaluation Methods

B.2.1 Testbed

The experimental testbed will be drawn from our fleet of existing mobile robots including: an actuated Hummer, 3 Denning Mobile Robots, 5 Nomad 150s, 4 Pioneer ATs, and a Hermes II. Missions will be designated for particular robots and tasked as needed. Section I describes our facilities in more details. Laptop workstations, as currently being developed in our effort for DARPA's Tactical Mobile Robotics Program will be used for the warfighter interface in the field.

B.2.2 Evaluation

Our evaluation methods will include: (1) Extensive simulation studies demonstrating the effectiveness of the new learning methods in DARPA specified mission scenarios, including performance and convergence studies; (2) An annual sequence of progressively more challenging demonstrations on the robotic testbed illustrating the strength and advantages of the proposed learning methods with a target capability designated for each year of the effort; and (3) Usability studies to determine the effectiveness of integrating machine learning capabilities over non-adaptive approaches by evaluating their ease-of-use for the design of relevant missions by various human personnel, including those with skills comparable to actual warfighter end-users (or the warfighters themselves if available).

It is crucial that metrics be developed early in the life of a development effort in order to ensure the utility of the final product. An evaluation process must also be constructed that is capable of measuring the performance of the product relative to those metrics, in order to guide its ongoing development and to ensure enduser acceptance. Usability studies will provide the primary vehicle for testing our research products, based upon our earlier experiences in this area [55,56].

A set of usability attributes will be defined that is captured in a usability criteria specification table. This will include data on the attribute in question, what values reflects that attribute, the current level of user performance, and worst-acceptable, target, and best-possible levels. Based on these data, a series of usability experiments will be constructed, each of which contains specific objectives regarding the evaluation of the attributes in the specification table.

Standard operating procedures for the administration of our usability studies include:

- Administration of the experiments by a third party to eliminate bias.
- A uniform introduction to the toolset to all subjects.
- Participants are sequenced through a series of tasks over several days as necessary to prevent tiredness from affecting performance.
- The subjects are isolated in a usability lab and are observed via one-way glass and video cameras.
- Computer logs are recorded for the entire session.

Statistical analysis of the resulting data can provide confirmation of the achievement of target level performance or provide feedback for the modification of the underlying software product.

A fundamental contribution that transcends our own laboratory's particular approach to mission specification is the development of metrics and methodologies to evaluate learning methods for mobile robot missions in general. Consequently, it is expected that the usability methods and metrics could be applied to other DARPA efforts funded under this BAA if deemed appropriate by Program management in option year efforts.

Using methods that have been developed within our laboratory [55,56] that are consistent with usability testing in general, we will evaluate the ease of use of the design of missions by various personnel, including those with skills comparable to actual MARS end-users. Whenever possible actual military personnel will be used for subjects (e.g., Army ROTC students or in collaboration with military bases within Georgia) to provide feedback for the iterative design necessary in perfecting a system that will ultimately be satisfactory to customer requirements.

B.2.3 Simulation studies

Analysis of the performance of a mission and its resistance to faults will be analyzed through extensive simulation studies within our laboratory's *MissionLab* software framework. This will enable the comparison between various learning/adaptation alternatives relative to non-adaptive methods for various mission taskings using metrics such as the speed of mission completion, the likelihood of success in the presence of risk, how sensitive the designed system is regarding interagent communication failure, and other related factors.

Mission design performance must also be measured, i.e., how well a particular plan satisfies the requirements of the task. Selection of a performance metric is important because these metrics are often in competition - i.e., cost versus reliability. Some potential metrics for MARS missions are:

- **Cost** - Accomplish the task for the minimum cost. Use of this metric will tend to reduce the cost of the system and minimize the number of robots used.
- **Time** - Accomplish the task in minimum time. This metric will lead to a solution calling for the maximum number of taskable machines that can operate without interference.
- **Energy** - Complete the task using the smallest amount of energy. This is appropriate in situations where fuel reserves are limited.
- **Reliability** - Accomplish the task in a manner that will have the greatest probability of completion even at the expense of time or cost.
- **Survivability** - Accomplish the task in a manner that is most likely to preserve individual assets.

The task metric can also be a numeric combination of several measurements. Whatever the metric is, it must be measurable, especially in simulation. In our previous research [20], time to complete the task was chosen as the primary performance metric. It is easily and accurately measurable and conforms to what is frequently thought of as performance. No claim is made however that this is the "best" metric; distance, reliability, survivability, energy consumption or some context-dependent combination may be more useful. Analysis of which metrics are best suited for MARS operations will be an important aspect of the research.

B.2.4 Robotic Field Tests

Finally, several of the developed mission scenarios will be fielded on our laboratory's robotic hardware (Section I). These tests will be conducted with targeted end-users, where feasible, in addition to development system personnel. For example, operational scenarios such as explosive ordnance disposal (EOD), countermine operations, countersniper, and building entry may be included in the suite of tests developed.

During the first year of the program, the emphasis will remain on simulations. In these simulations, it is possible to observe the performance of learning algorithms without the investment of time and other resources in the operation of mechanical systems. But proven integrated hardware platforms will exist for informal testing and validation as needed during this period. During the second year of the program, however, the development and test emphasis shifts to these existing laboratory robotic platforms that have already been used with the *MissionLab* system. These platforms, herein referred to as the *laboratory platforms* include both the five Nomadic 150 machines for indoor scenarios and the four Pioneer AT

machines of indoor/outdoor scenarios, supplemented by Denning robots, an actuated Hummer, and several other robots. The large complement of systems allows experimentation with either homogeneous or heterogeneous societies of robots. Early usability studies will also occur in the same timeframe, where the subjects will be tasked to implement the same demonstration scenarios or simplified versions thereof.

In the option years, the government-supplied robotic platforms will be integrated into the *MissionLab* system and used as the primary hardware testbed. This involves the development of a low-level software interface layer to translate the *MissionLab* protocols to those specified by the new platform. This effort has been completed with several commercial robot platforms and should be a straightforward process carrying little risk. Once complete, demonstration scenarios will be implemented on the new hardware. These scenarios may be identical to those performed on the laboratory platforms, or they may be extended to take advantage of either a) unique hardware capabilities of the government-supplied platforms, or b) new capabilities of the learning algorithms as enhanced in the option years.

References

[1] Albus, J., McCain, H. and Lumia, R., "NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)", NBS Tech. Note 1235, Nat. Bureau Standards, 1987.

[2] Arbib, M.A., "Schema Theory", in *The Encyclopedia of Artificial Intelligence*, Second Edition, ed. S. Shapiro, Wiley-Interscience, pp. 1427-1443, 1992.

[3] Arkin, R.C., "Path Planning for a Vision-based Autonomous Robot", *Proc. SPIE Conference on Mobile Robots*, Cambridge, MA, pp. 240-249, 1986.

[4] Arkin, R.C., "Towards Cosmopolitan Robots: Intelligent Navigation in Extended Man-made Environments", *Ph.D. Thesis*, COINS Tech. Report 87-80, Univ. of Massachusetts, 1987.

[5] Arkin, R.C., "Motor Schema-Based Mobile Robot Navigation", *International Journal of Robotics Research*, Vol. 8, No. 4, August 1989, pp. 92-112.

[6] Arkin, R.C., "Navigational Path Planning for a Vision-based Mobile Robot", *Robotica*, Vol. 7, pp. 49-63, 1989.

[7] Arkin, R.C.,"Towards the Unification of Navigational Planning and Reactive Control", working notes, *AAAI 1989 Spring Symposium on Robot Navigation*, Stanford, March 1989.

[8] Arkin, R.C., "The Impact of Cybernetics on the Design of a Mobile Robot System: A Case Study", *IEEE Transactions on Systems, Man, and Cybernetics*, 20:6 (1990), pp. 1245-1257.

[9] Arkin, R.C., "Integrating Behavioral, Perceptual, and World Knowledge in Reactive Navigation", *Robotics and Autonomous Systems*, 6 (1990b), pp. 105-122.

[10] Arkin, R.C., "Behavior-Based Robot Navigation for Extended Domains", *Adaptive Behavior*, Vol. 1, No. 2, 1992.

[11] Arkin, R.C., "Cooperation without Communication: Multi-agent Schema Based Robot Navigation", *Journal of Robotic Systems*, Vol. 9(3), April 1992b, pp. 351-364.

[12] Arkin, R.C., Balch, T., Collins, T., Henshaw, A., MacKenzie, D., Nitz, E., Rodriguez, R., and Ward, K., "Buzz: An Instantiation of a Schema-Based Reactive Robotic System", *Proc. Int. Conf. Intelligent Autonomous Systems: IAS-3*, Pittsburgh, 1993, pp. 418-427.

[13] Arkin, R. and MacKenzie, D., "Temporal Coordination of Perceptual Algorithms for Mobile Robot Navigation", *IEEE Trans. Robotics and Auto.*, Vol. 10, No. 3, June 1994, pp. 276-286.

[14] Arkin, R.C. and Murphy, R.R., "Autonomous Navigation in a Manufacturing Environment", *IEEE Transactions on Robotics and Automation*, Vol. 6, No. 4, August 1990, pp. 445-454.

[15] Arkin, R.C., Murphy, R., Pearson, M. and Vaughn, D., "Mobile Robot Docking Operations in a Manufacturing Environment: Progress in Visual Perceptual Strategies", *Proc. IEEE International Workshop on Intelligent Robots and Systems '89*, Tsukuba, Japan, pp. 147-154.

[16] Asada, M., Noda, S., Tawaratsumida, S., and Hosoda, K., "Vision-based Reinforcement Learning for Purposive Behavior Acquisition", *Proc. IEEE Inter. Conf. Rob. and Auto.*, 1995, pp. 146-153.

[17] Balch, T., *Behavioral Diversity in Learning Robot Teams*, Ph.D. Dissertation, College of Computing, Georgia Tech, 1998.

[18] Ashley, K., & Rissland, E. (1987). Compare and Contrast, A Test of Expertise. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, 273–284.

[19] Balch, T., "Integrating RL and behavior-based control for soccer", IJCAI-97 Workshop on Robocup, 1997.

[20] Balch, T. and Arkin, R.C., "Communication in Reactive Multiagent Robotic Systems", *Autonomous Robots*, Vol. 1, No. 1, pp. 27-52, 1994.

[21] Balch, T. and Arkin, R.C., "Motor Schema-based Formation Control for Multiagent Robot Teams", *Proc. 1995 International Conf. on Multiagent Systems*, San Francisco, CA, pp. 10-16.

[22] Balch, T., Boone, G., Collins, T., Forbes, H., MacKenzie, D., and Santamaría, J., "Io, Ganymede, and Callisto - A Multiagent Robot Trash-collecting Team", *AI Magazine*, 16:2 (1995), pp. 39-51.

[23] J. Blythe, "Planning with External Events", *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 1994, pp. 94–101.

[24] M. Bradakis, T. Henderson, and J. Zachary, "Reactive Behavior Design Tools", *Proc. IEEE International Symposium on Intelligent Control*, pp. 178-183, 1992.

[25] R. Brooks, "A Robust Layered Control System for a Mobile Robot", *IEEE Journal of Robotics and Automation*, Vol/. 2, no. 1, Mar. 1986, pp. 14-23.

[26] R.A. Brooks, "The Behavior Language: User's Guide", *MIT AI Memo 1227*, 1990.

[27] Cameron, J., MacKenzie, D., Ward, K., Arkin, R., and Book, W., "Reactive Control for Mobile Manipulation", *1993 International Conf. Robotics and Auto.*, pp. 228-235, 1993.

[28] Cheeseman, P. and Stutz, J., "Bayesian Classification (AutoClass): Theory and Results", in *Advances in Knowledge Discovery and Data Mining*, Kluwer, 1996.

[29] Chien, S.A., Gervasio, M.T., & DeJong, G. (1991). "On Becoming Decreasingly Reactive: Learning to Deliberate Minimally",*Proc. 8th Inter. Workshop on Machine Learning*, 288-292.

[30] Clark, R.J., Arkin, R.C. and Ram, A., "Learning Momentum: On-line Performance Enhancement for Reactive Systems", *Proc. 1992 IEEE International Conference on Robotics and Automation*, Nice, France, May 1992, pp. 111-116.

[31] Dean, T., Kaelbling, L, Kirman, J., and Nicholson, A., , "Planning under Time Constraints in Stochastic Domains", *Artificial Intelligence*, 76(1-2)35-74, 1995.

[32] DeJong, G., 1994, Learning to Plan in Continuous Domains. *Artif. Intel.*, 65(1):71-141.

[33] *Reasoning with Uncertainty in Robotics*, eds. Dorst, van Lambalgen, and Voorbraak, Lecture Notes in Artificial Intelligence, Vol. 1093, 1996, Springer.

[34] Elfes, A., "A Sonar-based Mapping and Navigation System", *Proc. IEEE International Conference on Robotics and Automation*, San Francisco, CA, pp. 1151-1156, 1986.

[35] Gat, E., "ALFA: A Language for Programming Reactive Robotic Control Systems", *Proc. 1991 IEEE Inter. Conf. Robotics and Auto.*, Sacramento, CA, pp. 1116-1120.

[36] Goel, A., Ali, K., Donnellan, M., Gomez de Silva Garza, A., and Callantine, T., "Multistrategy Adaptive Path Planning", *IEEE Expert*, Vol. 9, No. 6, December 1994.

[37] J. Gowdy, "SAUSAGES: A Framework for Plan Specification, Execution, and Monitoring", *SAUSAGES Users Manual*, Version 1.0, Robotics Institute, Carnegie Mellon, Feb. 1991.

[38] Grefenstette, J.J., & Ramsey, C.L. (1992). An Approach to Anytime Learning. In Sleeman & Edwards (eds.), *Machine Learning: Proc. 9th Inter. Conf.*, 189-195, Aberdeen, Scotland.

[39] P. Haddawy, A. Doan and R. Goodwin, "Efficient Decision-Theoretic Planning: Techniques and Empirical Analysis", *Proc. Conf. Uncertainty in Artificial Intelligence*, 1995, pp. 229–236.

[40] K. Haigh, *Situation-Dependent Learning for Interleaved Planning and Robot Execution*, Ph.D. Thesis, School of Computer Science, Carnegie Mellon University, 1998.

[41] Hammond, K.J., *Case-Based Planning: Viewing Planning as a Memory Task*. Perspectives in Artificial Intelligence. Academic Press, Boston, MA, 1989.

[42] Hix, D. and Hartson, H., *Developing User Interfaces*, 1993, Wiley.

[43] L. P. Kaelbling, "An Architecture for Intelligent Reactive Systems", *SRI International Technical Note 400* Oct. 1986.

[44] L. P. Kaelbling and S. J. Rosenschein, "Action and Planning in Embedded Agents", *Robotics and Autonomous Systems*, Vol. 6, 1990, pp. 35-48.

[45] S. Koenig and R.G. Simmons, "Unsupervised Learning of Probabilistic Models for Robot Navigation", *Proc. International Conference on Robotics and Automation*, pp. 2301-2308, 1996.

[46] S. Koenig and R.G. Simmons, "Passive Distance Learning for Robot Navigation", *Proc. International Conference on Machine Learning*, pp. 266-274, 1996.

[47] S. Koenig and R.G. Simmons, "Xavier: A Robot Navigation Architecture Based on Partially Observable Markov Decision Process Models", in *Artificial Intelligence Based Mobile Robotics*, ed. Kortenkamp, Bonasso and Murphy, 1998, MIT Press, pp. 91–122.

[48] S. Koenig, "Optimal Probabilistic and Decision-Theoretic Planning using Markovian Decision Theory", 1991, Tech. Rep., Computer Sci. Dept., Univ. Calif., Berkeley, UCB/CSD 92/685.

[49] Kolodner, J., Simpson, R., & Sycara, K. (1985). A Process Model of Case-Based Reasoning in Problem Solving. In A. Joshi, editor, *Proc. AAAI-85*, 284-290, Los Angeles.

[50] Kopeikina, L., Brandau, R., & Lemmon, A. (1988). Case-Based Reasoning for Continuous Control. In *Proc. Workshop on Case-Based Reasoning*, 250–259, Clearwater Beach, FL.

[51] J. Kosecka and R. Bajcsy, "Cooperative Behaviors - Discrete Event Systems based approach", 1993.

[52] N. Kushmerick, S. Hanks and D. Weld, "An Algorithm for Probabilistic Planning", *Artificial Intelligence*, 1995, 76(1–2), pp. 239–286.

[53] Lin, L., "Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching", *Machine Learning*, Vol. 8, pp. 293-321, 1992.

[54] Lyons, D. and Arbib, M., "A Formal Model of Computation for Sensory-based Robotics", *IEEE Transactions on Robotics and Automation*, Vol. 6, No. 3, June 1989, pp. 280-293.

[55] MacKenzie, D., "A Design Methodology for the Specification of Behavior-based Robotic Systems", *Ph.D. Dissertation*, College of Computing, Georgia Tech, Atlanta, 1996.

[56] MacKenzie, D. and Arkin, R., "Evaluating the Usability of Robot Programming Toolsets", *International Journal of Robotics Research*, Vol. 4, No. 7, April 1998, pp. 381-401.

[57] MacKenzie, D., Arkin, R.C., and Cameron, R., "Multiagent Mission Specification and Execution", *Autonomous Robots*, Vol. 4, No. 1, Jan. 1997, pp. 29-52.

[58] MacKenzie, D. and Balch, T., "Making a Clean Sweep: Behavior Based Vacuuming", in Working Notes, *1993 AAAI Fall Symposium: Instantiating Real-world Agents*, AAAI, Raleigh.

[59] Mahadevan, S. and Connell, J., "Automatic Programming of Behavior-based Robots using Reinforcement Learning", *Proc. AAAI-91*, Anaheim, July 1991, pp. 768-73.

[60] Meystel, A., "Planning in a Hierarchical Nested Controller for Autonomous Robots", *Proc. 25th Conf. Decision and Control*, Athens, Greece, 1986, pp. 1237-1249.

[61] Miller, J. and Jeffries, R., "Usability Evaluation: Science of Trade-offs", *IEEE Software*, Sept., 1992, 97-102.

[62] Mitchell, T.M. (1990). Becoming Increasingly Reactive. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 1051–1058, Boston, MA.

[63] Murphy, R.R., and Arkin, R.C., "SFX: An Architecture for Action-Oriented Sensor Fusion", *Proc. 1992 Inter. Conf. on Intelligent Robotics and Systems*, Raleigh, N.C., July 1992.

[64] Murphy, R.R., "An Application of Dempster-Shafer Theory to a Novel Control Scheme for Sensor Fusion", *Proc. SPIE Stochastic Methods in Signal Processing, Image Processing, and Computer Vision*, San Diego, CA, July 1991.

[65] L. Parker, "Heterogeneous Multi-Robot Cooperation", *PhD Dissertation*, Department of Electrical Engineering and Computer Science, MIT, 1994.

[66] Ram, A. (1993). Indexing, Elaboration & Refinement: Incremental Learning of Explanatory Cases. *Machine Learning*, 10:201–248.

[67] Ram, A., Arkin, R.C., Moorman, K., and Clark, R., "Case-based Reactive Navigation: A Case-based Method for On-line Selection and Adaptation of Reactive Control parameters in Autonomous Robotic Systems", *IEEE Transactions on Systems, Man, and Cybernetics*, 1997.

[68] Ram, A., Arkin, R., Boone, G., and Pearce, M., "Using Genetic Algorithms to Learn Reactive Control Parameters for Autonomous Robotic Navigation", *Journal of Adaptive Behavior*, Vol. 2, No. 3, pp. 277-305, 1994.

[69] A. Ram & J.C. Santamaria (1997), "Continuous Case-Based Reasoning", *Artificial Intelligence*, (90)1-2:25–77.

[70] P.J. Ramadge and W.M. Wonham, "The control of discrete event systems", *Proceedings of the IEEE*, Jan. 1989, vol. 77-1, no. 1, pp. 81-97.

[71] J. Rosenblatt and D. Payton, "A Fine-Grained Alternative to the Subsumption Architecture for Mobile Robot Control", *IEEE INNS Intern. Conf. Neural Networks*, 1989, (2), pp. 317-323.

[72] A. Saffiotti, K. Konolige and E. Ruspini, "A Multivalued Logic Approach to Integrating Planning and Control", *Technical Report 533*, SRI AI Center, Menlo Park, California, 1993.

[73] Saridis, G. and Valvanis, K., "On the Theory of Intelligent Controls" *Proc. SPIE Conf. on Advances in Intelligent Robotic Systems*, pp. 488-495, 1987.

[74] S. Schneider, V. Chen and G. Pardo-Castellote, "The ControlShell Component-Based Real-Time Programming System", *Proc. IEEE Int. Conf. Rob. and Auto.*, 1995, pp. 2381-2388.

[75] Shen, W., *Autonomous Learning from the Environment*, 1994, Computer Science Press.

[76] D. Stewart and P. Khosla, "Rapid Development of Robotic Applications using Component-Based Real-Time Software", *Proc. Intelligent Robotics and Systems*, 1995, pp. 465-470.

[77] Stroulia, E., "Reflective Self-Adaptive Systems", *Ph.D. Thesis*, College of Computing, Georgia Institute of Technology, Atlanta, GA, 1994.

[78] Thrun, S., "An Approach to Learning Mobile Robot Navigation", *Robotics and Autonomous Systems*, Vol. 15, No. 4, 1995, pp. 301-320.

[79] Veloso, M., & Carbonell, J.G. (1993). Derivational Analogy in PRODIGY: Automating Case Generation, Storage, and Retrieval. *Machine Learning*, 10(3):249–278.

[80] Watkins, C. and Dayan, P., "Q-Learning", *Machine Learning*, Vol. 8, pp. 279-92, 1992.

C. Deliverables

The deliverables associated with each of the Tasks specified in the Statement of Work (Section D) appears below.

- Task 1. Learning Momentum for Adaptation

  1. Situational analyses of relevant military scenarios including the identification of salient environmental features and the associated behavioral modifying rulesets.
  2. Software capable of automatic classification methods for situational extraction and recognition for use in DARPA specified test scenarios.
  3. Learning momentum software, fully integrated into the *MissionLab* Autonomous Robot Software environment, that continuously adjusts behavioral gains and that is capable of adapting to a significant range of battlefield situations and missions.
  4. An easy-to-use usability validated graphical interface, enabling an average warfighter to deploy this learning capability with minimal effort.
  5. Demonstrations of this capability in simulated, laboratory, and, where appropriate, DARPA-specified field exercises.

- Task 2. Case-Based Reasoning for Behavioral Selection

  1. Analysis of requirements for strategic guidance from cases, indicating the appropriate outputs of the case-based reasoning system under a variety of task demands for the *MissionLab* system.
  2. Analysis of applicability conditions for cases, indicating the appropriate environmental triggers for retrieving and deploying cases.
  3. Implementation of continuous case-based reasoning techniques, based on the SINS system, and their complete integration into the *MissionLab* software environment.
  4. Empirical evaluation of the implemented techniques through extensive simulation and robotic studies within the context of the *MissionLab* system.

- Task 3. Q-learning for Behavioral Assemblage Selection

  1. Analysis of DARPA relevant scenarios in terms of suitable I/O state specifications.
  2. Development and integration of reinforcement learning software within the *MissionLab* framework.
  3. Integration into *MissionLab* user interface and subsequent usability validation of effectiveness.
  4. Evaluation of the implemented techniques through extensive simulation and robotic studies within the context of the *MissionLab* system in relevant DARPA mission scenarios.

- Task 4. CBR Wizardry

  1. Design of case representation and selection mechanisms appropriate for DARPA scenarios.
  2. Development of automated case acquisition tools.

3. Wizard-style interactive interfaces to guide end-users easily through mission specification and planning based on the results of prior experience.
4. Evaluation via usability studies of the effectiveness of these methods on average users, including military personnel where feasible.
5. Demonstrations of this capability in simulated, laboratory, and, where appropriate, DARPA-specified field exercises.

- Task 5. Probabilistic Planning and Execution

1. Developed test environments for suitable military scenarios.
2. An integrated probabilistic learning/planning system, as an enhancement to the existing *MissionLab* system.
3. Empirical evaluation of the benefits of the implemented modules through extensive simulation and robotic studies in DARPA relevant missions.

- Task 6. *MissionLab* Interface to Government-supplied Platform

1. Integration plan
2. Low-level interface software layer (option years)
3. Updates to *MissionLab* documentation

- Task 7. Demonstration Development and Support

1. Demonstration plan
2. Simulations of all learning algorithms (described above)
3. Release of new *MissionLab* version
4. Robotic hardware demonstrations of each learning algorithm on laboratory platforms
5. Robotic hardware demonstrations of each learning algorithm on government-supplied platforms (option years)

- Task 8. Program Administration

1. Monthly cost and status reports
2. Technology transfer plan
3. Interim progress reports (presentations)
4. Final report
5. Other required DARPA reports as outlined in contract (e.g., Subsystems Specification, Interface Control Document)

D. Statement of Work

There are 8 Major tasks associated with this research:

- Research and Development for Adapting Reactive Control within a Hybrid MARS Architecture
  1. Learning Momentum for Adaptation
  2. Case-based Reasoning for Behavioral Selection
  3. Q-learning for Behavioral Assemblage Selection
- Research and Development for Learning in support of Deliberative Planning within a Hybrid MARS Architecture
  4. Case-based Reasoning Wizardry
  5. Probabilistic Planning and Execution
- General Tasks
  6. *MissionLab* Port to Government-supplied Platform
  7. Demonstration Development and Support
  8. Program Administration

The specific work descriptions for each task are presented in the following sections.

D.1 Learning Momentum for Adaptation

- DARPA-relevant scenarios will be constructed in coordination with the sponsor. The domains will be analyzed in terms of salient perceptual features pertinent to the successful execution of a robotic mission. These will be classified into various situations that will serve serve as the arena for exercise of the learning methods.
- Situationally dependent rulesets will be created and tested in simulation. The system will be integrated within the *MissionLab* software environment.
- The results will be ported to our laboratory's mobile robots and evaluated in terms of mission performance.
- Usability studies will be conducted to verify the utility of the interface and to provide guidance for iterations in the design of the interface and ruleset in the contract option years.

D.2 Case-based Reasoning for Behavioral Selection

- We will create example tasks in the *MissionLab* architecture, and through empirical studies with the system we will determine common patterns of behaviors which should be encapsulated as cases by the adaptation and learning subsystem. We will then analyze these patterns to determine the best representation for the cases, including what the internals of the case will look like, what recommendations the case will provide to the system, and what environmental triggers should the case be indexed under.

- We will extend our previous case-based reasoning algorithms to use the new representations and implement them in the *MissionLab* sysyem. The algorithms will be tested using example scenarios based on the empirical studies mentioned above. We will compare the output of the algorithms with the patterns identified previously to determine how well they are performing in the new system. Based on this comparison, we will modify the algorithms as necessary to ensure that they are performing the adaptation and learning process appropriately.
- When the algorithms are implemented and tested, we will run extensive simulation studies, followed by studies with actual robots, to demonstrate that the system is in fact performing as expected under a variety of environmental situations and task descriptions.

D.3 Q-learning for Behavioral Assemblage Selection

- Integration of Q-learning into the run-time execution system of *MissionLab* will be performed by incorporating real-time performance evaluation methods to ensure accurate reinforcement at the correct time. This will require analysis of the underlying representations used within *MissionLab* to provide efficient operation with these methods.
- DARPA-specified scenarios will be utilized to evaluate the performance enhancement gained via Q-learning quantitatively.
- Real robots will be used in field experiments each year for proof of concept.
- Extensions of the underlying methods into mechanisms beyond behavioral assemblage selection will be investigated in the option years 3 and 4.
- Usability studies will be undertaken to evaluate the utility of these methods by end-users.

D.4 Case-based Reasoning Wizardry

- User interface design from both a user and task perspective will be undertaken to ensure ease-of-use in the end product.
- Case representations will be selected and then automatic case acquisition methods will be implemented within *MissionLab* with extensive user help and guidance facilities.
- Using DARPA-specified missions we will evaluate quantitatively (first in simulation then on actual robots) the value of these methods in the planning stages of mission design.
- Usability studies will be conducted to provide an understanding of how deployable these methods are, and what enhancements should be made to the system in the option years.

D.5 Probabilistic Planning and Execution

- Using the the *MissionLab* architecture, we will create several prototypical environments and navigation tasks, and then perform extensive empirical studies with the existing navigation behaviors in these environments.
- We will analyze the results of these experiments and, based on the experience gained, analyze the requirements of the learning, planning, and navigation modules and determine how to classify the environment into categories.
- We will develop stand-alone versions of the learning, planning, and navigation modules and test them in simulation using the prototypical environments and navigation tasks.
- We will integrate the modules into the *MissionLab* system and test them on real robots, refine them to increase their robustness and performance, and quantify the improvement achieved over the current system.

D.6 *MissionLab* to Government-supplied Platform

During the base program period, we will make a preliminary assessment of any proposed government-supplied platform and develop a detailed plan for the integration of this platform into the *MissionLab* system. In the first option year, this plan will be executed, and a complete software interface will be developed. In the second option year, all learning algorithms will be executed in conjunction with the integrated government-supplied platform, making any required modifications to ensure the reliable demonstration of these algorithms under task 7.

D.7 Demonstration Development and Support

As partially described in the learning-related tasks above, demonstrations will be developed a) in simulation, b) for the laboratory platforms (actual robots), and c) for the government-supplied platforms. Prior to this, a demonstration plan will be developed, taking into account military interests in various scenarios, along with the expected physical capabilities of the actual robots. During the course of the development of the learning algorithms, the *MissionLab* system will be modified as necessary to support these algorithms and the requirements of the demonstration scenarios. This includes the maintenance of the software configuration and the controlled release of at least one new version.

D.8 Program Administration

Graduate Research Assistants will be monitored in the development of the learning algorithms, and supervisory faculty will be constantly in contact with the primary project director (Arkin). The learning algorithm development tasks will be coordinated so that their results can be effectively integrated into *MissionLab*, including their execution on actual robots.

Regular meetings (usually weekly) of affiliated faculty and student researchers will be conducted to update status and share potentially useful information. Monthly cost and status reports will be provided, along with interim progress reports at review meetings as required by the government. A Technology Transfer Plan will be developed, as well as other required documents as specified in the final contract.

F. Technology Transfer

The technologies developed under the proposed program that are transferable will include:

1. Algorithms – Learning Momentum for Adaptation
2. Algorithms – CBR for Behavioral Selection
3. Algorithms – Q-learning for Behavioral Selection
4. Algorithms – CBR Wizardry
5. Algorithms – Probabilistic Planning and Execution
6. Human/Computer Interface – *MissionLab* GUI and usability results
7. Robot Control – Interface to government-supplied platform
8. Mission Specification and Robot Tasking– Scenario development and techniques

Georgia Tech has a strong record in the transfer of *MissionLab* technology to commercial and government institutions. Since its first release, *MissionLab* has been publicly available at the Web site of the Mobile Robotic Laboratory at http://www.cc.gatech.edu/aimosaic/robot-lab/research/MissionLab.html. During the course of the DARPA Demo II program, *MissionLab* was not only integrated by Georgia Tech with products from other institutions, but also was successfully adopted *by other institutions* for their own developmental needs, as illustrated by surveillance applications developed by the University of Texas at Arlington.

The free availability of *MissionLab* for research and educational purposes does not preclude the development of customized or proprietary extensions to the system, including the incorporation of the system into classified products by other performers. Furthermore, no additional restrictions will be placed upon the software components developed during enhancement of *MissionLab* as described for this research. The new release(s) of the system as described herein will be available as in the past. This includes the low-level software interface for the government-supplied robotic platforms, thus ensuring that other contractors will be able to use these platforms and *MissionLab* "out of the box." Furthermore, during the early stages of the program, other contractors who have access to platforms supported by *MissionLab* (including Nomadics 150, Pioneer AT, and Denning robots) will be able to use the results of this research into learning algorithms with their own robots, without the need to wait for the delivery of government-supplied platforms.

In accordance with government review and release requirements, the results of the learning algorithm development will be published at relevant conferences and in technical journals. This further assists the transfer of this technology to other sectors. A detailed Technology Transfer Plan will be developed early in the program to more completely define this process, including an assessment of any associated risks.

If desired by the government, a dedicated Web site can be maintained for this program, similar to that created for the DARPA Tactical Mobile Robotics (TMR) program. Like the TMR site, access can be "hidden" from casual viewers, or more secure access procedures can be established in accordance with governmental requirements. The availability of monthly reports, briefings, and other contractual documents over this site can further facilitate technology transfer. It is also possible to execute actual simulations over the Web, as is being done with the TMR site. (http://www.cc.gatech.edu/ai/robot-lab/tmr).

G. Related Research

G.1 Q-learning

Q-learning was used at IBM [59] to teach a behavior-based robot how to push a box. The learning problem consisted of deciding which of 5 possible actions to take for any of the the approximately one-quarter million perceptual states that results in efficient finding and pushing of boxes around a room without getting stuck. The behavioral controller consisted of three behaviors. It was observed that box-pushing was substantially improved using Q-learning over a random agent. It was also compared to a hand-coded and tuned behavioral controller. The importance of this work lies in its empirical demonstration of the feasibility of Q-learning as a useful approach to behavior-based robotic learning.

Researchers at Osaka University [16] applied vision-based reinforcement learning to the task of shooting a ball into a goal. They used Q-learning within their system. The set of states for the utility function $Q(x, a)$ was defined in terms of the input visual image. The action set consisted of the inputs for each of the two independent motors which power each wheel. Easy situations were devised by the trainer to first allow the system to improve its performance in this constrained situation. As this is mastered, more challenging states are introduced. This improves learning performance convergence dramatically. The system was implemented on a tracked robot. Looking at the final action-state data, approximately 60% of the stimulus-response mappings were correctly created. Considering that the robot had no ability whatsoever to get the ball into the goal initially, these results are encouraging.

Thrun demonstrated hybrid learning including Q-learning for robot navigation [78]. Its particular task is to recognize and then move to a specific target. Navigation consists of a sequence of specified actions, rather than being controlled by a set of active behaviors. A learning episode consists of the robot starting at some point within the lab and terminating when either the robot was directly in front of the target (rewarded) or the target left the field of view (penalized). Q-learning is used to determine the action policies. The net effect was successful and relatively rapid on-line learning and navigation for this particular task.

G.2 Case-based Learning and Reasoning

Case-based reasoning (CBR) systems have traditionally been used to perform high-level reasoning in problem domains that can be adequately described using discrete, symbolic representations. For example, CHEF uses case-based planning to create recipes [41], AQUA uses case-based explanation to understand newspaper stories[66], HYPO uses case-based interpretation for legal argumentation [18], MEDIATOR uses case-based problem solving for dispute resolution [49], and PRODIGY uses CBR in the form of derivational analogy for high-level robot planning [79].

We have been investigating the problem of performance and learning in continuous, real-time problem domains, such as autonomous robotic navigation. Continuous problem domains require different underlying representations and place additional constraints on the problem solving process [32]. We are developing new methods for CBR which can be used to guide action and to learn in continuous problem domains. The research presented here also has implications for the design of CBR systems in general.

Continuous CBR is a variation of traditional CBR that can be used to perform continuous tasks. The underlying steps in the method are similar, namely, situation assessment, case retrieval, adaptation and execution, and learning. The ACBARR and SINS systems perform a kind of case-based planning, and in that respect are similar to CHEF [41]. However, there are several interesting differences due to the continuous nature of the domain, and to the

on-line nature of the performance and learning tasks. Our approach is also similar to [50]'s use of case-based reasoning for real-time control. Their system, though not intended for robotics, is designed to handle the special issues of time-constrained processing and the need to represent cases that evolve over time. They suggest a system that performs the learning task in batch mode during off peak hours. In contrast, our approach combines the learning capabilities of CBR with on-line, real-time aspects of reactive control. In this respect, our research is also different from earlier attempts to combine reactive control with other types of higher-level reasoning systems (e.g., [29,62]), which typically require the system to "stop and think." In our systems, the perception-action task and the adaptation-learning task are integrated in a tightly knit cycle, similar to the "anytime learning" approach of [38].

G.3 Probabilistic Learning in Robotic Systems

AI researchers have often studied probabilistic planning techniques in simulated simplified robot navigation domains. Some approaches assume that they only have to select macro actions that move the robot directly to its destination. These planners are based on a variety of techniques, from symbolic planning techniques augmented with probabilities (including non-linear planning and refinement planning) to probabilistic reasoning techniques (including Bayesian networks and Markov Decision Processes). Probabilistic planners from AI include [48], [23], [52], [39], and [31]. Probabilistic planning in robotics is often used to integrate path planning and robot navigation tightly, to make sure that the robot is able to follow a plan despite pose uncertainty during navigation. This has been done using Kalman filters, Bayesian networks, and (a recently very popular approach) Partially Observable Markov Decision Process models. An overview on probabilistic planning approaches in robotics appears in [33].

Map learning approaches and model learning approaches (actuator models, sensor models, as well as other parameters) are common in robotics. Higher-level learning approaches have been used less frequently. Approaches that learn a model of the environment as well as the costs of applying actions in the environment include [75] and [40].

G.4 Specification Languages

The configuration definition language (CDL) within *MissionLab*, differs from comparable languages. First, it is architecture and robot independent. This permits construction of configurations which transcend individual robot constraints. Second, it incorporates recursive combination mechanisms to facilitate information hiding and construction of high-level primitives. Finally, it relies on explicit hardware binding to prevent hardware issues from diffusing into the behavioral design.

REX/Gapps: The REX/Gapps architecture[43] partitions perception from action. A REX program is compiled into a synchronous digital circuit which can be executed to implement the specified system. Gapps[44] is used to specify the goal-oriented planning component for a robot. Neither REX nor Gapps supports information hiding as does CDL in *MissionLab*. Methods for activating/deactivating modules as their utility changes are absent.

RS: Robot Schemas (RS) [54] is based on the port automata model of computation using synchronous communication. The assemblage mechanism facilitates information hiding, modularity, and incremental development. RS does not provide mechanisms for expressing coordination between multiple robots cooperating on a task.

Discrete Event Systems Theory (DES): DES [70] models systems as finite state automata with inputs as *observations* and outputs are *actions*. The perception-action cycle is

broken into discrete *events* where an event is an abrupt transition in the system state. A small robotic team architecture [51] has been developed using DES theory.
Multivalued Logic: Multivalued logic provides a mechanism capable of supporting formal analysis for combining behaviors. Techniques are presented [72] for formally describing behaviors being combined in various fashions. The application of multivalued logic to planning requires an explicit high-fidelity model of the environment to support analysis.
ALFA: ALFA [35] (A Language For Action), is used to describe behavior-based robots using a LISP-like syntax. ALFA has been used to specify the reactive execution component of a hybrid architecture (ATLANTIS). ALFA is more of an operating system and design methodology than just a language for describing mobile robot behaviors.

G.5 Architectures

Subsumption and its variants: The subsumption architecture [25] has been used to construct a range of mobile robots. However, the subsumption architecture remains more of a design philosophy than a formal specification. The Behavior Language[26] is the LISP-based parallel programming language used to specify subsumption configurations. Subsumption is often unwieldy and difficult to debug as new layers are added. There is no mechanism to support information hiding since the layering scheme is transparent. The subsumption architecture has served as a basis for the ALLIANCE architecture for controlling a heterogeneous robots [65].
UGV related architectures: The DAMN architecture [71] converts behaviors into a collection of simple transfer functions which are not allowed to contain state information. SAUSAGES[37] provides a behavior configuration specification language as well as run-time execution and monitoring support. What SAUSAGES does not provide is abstraction. CDL facilitates abstraction through recursive composition of components into new components. A variant of SAUSAGES called MRPL is used in the ARPA UGV robots. SAUSAGES is currently supported as a target architecture from CDL.

G.6 Graphical Programming Tools

Khoros: Khoros is a powerful system for graphically constructing and running image processing tasks from a collection of primitive operators. The user selects items from a library of procedures and places them on the work area as icons (glyphs). Connecting dataflows between glyphs completes construction of the program. Each glyph represents a distinct program which is instantiated as a separate process at run-time. This idea has been extended to allow the recursive specification of robot control programs in the *MissionLab* toolset.
Onika: Onika [76], from CMU, is optimized for the rapid graphical construction of control programs for robot arms. Programs are constructed by placing a linear sequence of icons using a puzzle piece metaphor. Onika includes a simulation system for evaluating control programs targeted for robot arms, but it does not include support for simulating or commanding mobile robots. The style is reminiscent of electronic schematic diagrams.
ControlShell: ControlShell[74] is a commercial graphical programming toolset from Real-Time Innovations used to construct complex real-time systems. It is based on Onika and has a similar schematic look and feel. The lack of support for recursive construction in ControlShell limits reuse and information hiding in complicated designs. ControlShell does not support explicit hardware binding or support any runtime architectures but its own.
GI Joe: The GI Joe toolset[24] allows graphical construction and visualization of finite state automata. Overall, GI Joe provides support for FSAs but little else.

H. Key Personnel

Principal Investigator and Project Director: Ronald C. Arkin

Ronald C. Arkin received the B.S. Degree from the University of Michigan, the M.S. Degree from Stevens Institute of Technology, and a Ph.D. in Computer Science from the University of Massachusetts, Amherst in 1987. He then assumed the position of Assistant Professor in the College of Computing at the Georgia Institute of Technology where he now holds the rank of Professor and is the Director of the Mobile Robot Laboratory. Dr. Arkin's research interests include reactive control and action-oriented perception for the navigation of mobile robots and unmanned aerial vehicles, robot survivability, multiagent robotic systems, and learning in autonomous systems. He has over 90 technical publications in these areas. Prof. Arkin has recently completed a textbook entitled *Behavior-Based Robotics* published by MIT Press in the Spring of 1998 and has co-edited (with G. Bekey) a book entitled *Robot Colonies* published by Kluwer in the Spring of 1997. Funding sources have included the National Science Foundation, DARPA, U.S. Army, Savannah River Technology Center, and the Office of Naval Research. Dr. Arkin serves/served as an Associate Editor for IEEE Expert and the Journal of Environmentally Conscious Manufacturing, as a member of the Editorial Boards of Autonomous Robots and the Journal of Applied Intelligence and is the Series Editor for the new MIT Press book series entitled *Intelligent Robotics and Autonomous Agents.* He is a Senior Member of the IEEE, and a member of AAAI and ACM. Dr. Arkin was elected to a three year term to the IEEE Robotics and Automation Society's Administrative Committee beginning in January 1999. A full vita, listing relevant publications, for Prof. Arkin is available at http://www.cc.gatech.edu/aimosaic/faculty/arkin/vita.html.

Co-Investigator: Ashwin Ram

Ashwin Ram is an Associate Professor in the Intelligent Systems Group of the College of Computing of the Georgia Institute of Technology, an Associate Professor of Cognitive Science, an Adjunct Professor in the School of Psychology, and a faculty member in the Graphics, Visualization, and Usability Center, and the Mobile Robot Lab. He received his B. Tech. in Electrical Engineering from the Indian Institute of Technology, New Delhi, in 1982, and his M.S. in Computer Science from the University of Illinois at Urbana-Champaign in 1984. He received his Ph.D. degree from Yale University for his dissertation on "Question-Driven Understanding: An Integrated Theory of Story Understanding, Memory, and Learning" in 1989.

Dr. Ram's research interests lie in the areas of machine learning, case-based reasoning, and cognitive science, and he has several research publications in these areas in major journals and conferences. His work on multistrategy learning in intelligent agents and robotic systems has been published in major journals (including Artificial intelligence, Cognitive Science, Adaptive Behavior, and IEEE Systems Man and Cybernetics) and conferences (including AAAI, ICML, and CogSci). He is a co-editor of a book on Goal-Driven Learning, published by MIT Press/Bradford Books.

Dr. Ram is a member of the editorial boards of The Journal of the Learning Sciences and the International Journal of Applied Intelligence, and an Associate of Brain and Behavioral Sciences. He was co-chair of the 1994 Annual Conference of the Cognitive Science Society, the 1994 AAAI Spring Symposium on Goal-Driven Learning, the 1995 AAAI Fall Symposium on Adaptation of Knowledge for Reuse, the 1996 FLAIRS Special Track on Real-World Natural Language Understanding, and the 1998 International Conference on the Learning Sciences.

Dr. Ram's research has been supported by the National Science Foundation (NSF), the Air Force Office of Sponsored Research (AFOSR), the Army Research Lab (ARL), the Office of Naval Research (ONR), the EduTech Institute, Digital Equipment Corporation (DEC), and Yamaha Motor Corporation.

Co-Investigator: Sven Koenig

Sven Koenig joined the College of Computing at Georgia Tech as an assistant professor in 1998 after receiving a Ph.D. degree in computer science from Carnegie Mellon University. As part of his Ph.D. thesis "Goal-Directed Acting with Incomplete Information," he developed novel methods for decision making in high-stake decision situations, for interleaving planning and acting, and for robot navigation. Dr. Koenig also holds M.S. degrees from the University of California at Berkeley and Carnegie Mellon University, and is the recipient of various awards and fellowships, including the Tong Leong Lim Pre-Doctoral Prize of the University of California at Berkeley. He chaired the AAAI-97 Workshop On On-Line Search, chairs the AAAI-98 Spring Symposium on Search Techniques for Problem Solving under Uncertainty and Incomplete Information as well as the Student Abstract and Poster Session at the 1998 National Conference on Artificial Intelligence. He has been on the program committees of conferences such as the International Conference on Artificial Intelligence Planning Systems (AIPS) and the National Conference on Artificial Intelligence (AAAI), has given over 20 invited talks at research institutions nationwide, and published more than 25 full papers.

Sven Koenig's research centers around techniques for planning and learning that enable situated agents, such as robots, to act intelligently in their environments and exhibit goal-directed behavior in real-time, even if they have only incomplete knowledge of their environment, limited or noisy perception, imperfect abilities to manipulate it, or insufficient reasoning speed. He has, for example, developed the POMDP-Based Robot Navigation Architecture that uses partially observable Markov decision process models to provide a uniform framework for modeling actuator and sensor noise and for integrating topological information about the environment with approximate metric information. This architecture is now under investigation at several universities worldwide.

Senior Research Engineer: Thomas R. Collins

Dr. Thomas R. Collins is a Senior Research Engineer in the Electronic Systems Laboratory at the Georgia Tech Research Institute, with recent shared appointments in the School of Electrical Engineering and the Office of Interdisciplinary Programs. He received a Bachelor's degree in Mechanical Engineering in 1980, a Master of Science in Electrical Engineering in 1982, and a Ph.D. in Electrical Engineering in 1994, all from the Georgia Institute of Technology. He graduated first in his undergraduate class and received an NSF fellowship. His areas of expertise are mobile robots, robotic manipulators, hardware/software architectures for parallel computation in intelligent machine applications, modeling and simulation, and high-performance computer architectures.

Immediately prior to returning to Georgia Tech in 1984, Dr. Collins had been employed in the Manufacturing Systems Products division of IBM in Boca Raton, Florida, where he was a member of the design team responsible for the IBM 7565 manipulator, a large, high-accuracy hydraulic/electric robot targeted at precision assembly tasks. At Georgia Tech, Dr. Collins has been responsible for a wide range of technical areas, including digital hardware design, robotics, software development (both system and application), image processing techniques, computer graphics, computer simulation, and parallel processing techniques. Some

of this has fallen within the scope of two major contracts with the U. S. Army Strategic Defense Command. Much of the government research work has concentrated on hardware and software aspects of parallel-processing systems, including an expandable 32-processor crossbar-based parallel processor, digital simulation of combined continuous/discrete systems, and use of the Transputer in image synthesis and processing. Most recently, his technical responsibilities have included robotics, CCD imaging systems, radar signal processing, visualization, telecommunication, sensor technologies, and hardware/software integration for an unmanned aerial and ground vehicles. Funding sources have included the US Army, Air Force, the Department of Energy, DARPA, the Georgia Department of Transportation, Lucent Technologies, and BellSouth. He has participated with four student teams in AAAI mobile robot competitions, including winning entries in the 1994 and 1997 events. He has authored or co-authored over two dozen publications and technical reports and is a member of the IEEE and ACM.

Mobile Intelligence Inc. Subcontract: Douglas MacKenzie

Dr. MacKenzie received the B.S. degree in Electrical Engineering in 1985 and the M.S. degree in Computer Science in 1989, both from Michigan Technological University. He worked for two years as a Software Engineer in Allen-Bradley's Programmable Controller Division. In March 1997, Dr. MacKenzie graduated with a Ph.D. in Computer Science from the Georgia Institute of Technology. While at Georgia Tech, Dr. MacKenzie was heavily involved in the development of the *MissionLab* toolset and created the graphical Configuration Editor which brings visual programming to the behavior-based robotics domain. Dr. MacKenzie designed and conducted usability tests on *MissionLab*. Dr. MacKenzie also helped develop an integrated mobile manipulator for material handling which demonstrated the ability to autonomously transfer samples from one drum to a second drum in unstructured environments.

Mobile Intelligence was founded in 1993 by Dr. MacKenzie to conduct leading edge research and development in robotics and intelligent systems. Dr. Douglas C. MacKenzie recently served as the Principal Investigator for Mobile Intelligence's first government contract, DAAH01-97C-R302, a DARPA SBIR project to explore adding a speech command and control interface to a mobile robot using a COTS speech engine.

Since founding Mobile Intelligence, Dr. MacKenzie has designed the Cardinal and Blackbird robots: small, wheeled, indoor platforms which include a vision system, ultrasonic sensors, and an embedded PC. Dr. MacKenzie also has developed the Swarm robot control system to control these robots.

I. Facilities

In addition to the extensive equipment base, networking capabilities, and support provided within the College of Computing, the Mobile Robot Laboratory has specific resources dedicated to this and other related projects. Its facilities include:

- Robots: 4 Pioneer AT Mobile Robots; 2 Denning MRV-II Mobile Robots; 1 Denning DRV-I Mobile Robot; 5 Nomad 150 Mobile Robots; 1 Hermes Hexapod Robot; 1 Robotic actuated AM General Hummer with GPS/INS; 1 CRS+ A251 5DOF robot arm.
- Computer Workstations: 3 Sun Sparcstation IPCs; 2 Sun Sparcstation 5s; 1 Sun Sparcstation 20; 2 SGI Indy; 3 Dell Latitude CPi laptops; 1 SparcBook 3 Laptop; 5 Toshiba PC Laptops; 3 RDI Sun Laptops with vision boards; 1 Zenith PC; 1 Decstation 5000/120.
- Vision Hardware: 1 Sensar PV-1 Pyramid Image Processor; 1 Teleos AV100 Image Motion Processor; 6 NewtonLabs Cognachrome vision system; 10 CCD pulnix/cohu/sony cameras; 1 Pulnix low lux image-intensified video camera; 3 Directed perception pan/tilt platforms; 1 Zebrakinesis pan/tilt platform.
- Other Sensors: 1 Denning laser scanner;) 1 RWI Laser range scanner; 3 Electronic compasses; Lasiris laser striping system with Watec video camera.
- Communications: Radio links (10 freewaves, 2 proxims, 6 lawns)

Figure 6: Several of the Georgia Tech Mobile Robot Laboratory's Robots