Learning Momentum: On-line Performance Enhancement for Reactive Systems

Russell J. Clark, Ronald C. Arkin, and Ashwin Ram

Technical Report GIT-CC-91/37

College of Computing

Georgia Institute of Technology

Atlanta, GA 30332-0280

Abstract

We describe a reactive robotic control system which incorporates aspects of machine learning to improve the system's ability to successfully navigate in unfamiliar environments. This system overcomes limitations of completely reactive systems by exercising on-line performance enhancement without the need for high level planning. The results of extensive simulation studies using the learning enhanced reactive controller are presented.

1. Introduction

Reactive robotic control systems [1,5,11] have produced significant results in generating intelligent robotic action when compared to previous efforts. These systems typically decompose actions into behaviors in order to produce rapid real-time sensory response. How these systems can adapt ongoing behaviors to the environment is an important first step in addressing learning in reactive control. The particular approach described in this paper 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.

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 would still be 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 has concentrated 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. A simple example is that if you have been in an open area for a good while and have not encountered any obstacles, pick up your speed a bit. If, on the other hand, you are in a highly cluttered area and you are having a significant number of close calls, you would want to adjust your velocity downward or treat obstacles more seriously. 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.

In this paper, we describe a reactive robotic controller with the ability to adapt and improve a mobile robot's performance. This research builds on the schema control system based on our previous work[1] which is summarized in section 3. Section 4 outlines a rule-based adaptive system capable of adjusting schema control parameters and gain values and improving the reactive system's ability to successfully navigate in complex environments. 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.

2. Related Work in Robotic Learning

This work takes inspiration from other recent work combining the ideas of machine learning with robotic control. Much of this work is not based on reactive control systems but is aimed at systems which improve navigational ability by learning environmental features[12] or developing models of actions in the environment[7].

Bennett described a robotic arm controller based on an Explanation Based Learning (EBL) system which uses approximations to deal with inconsistencies between reactive rules and real-world experience[4]. Mitchell's Theo-Agent architecture is an EBL system which learns to be more reactive by generating rules from explanations of failure[10]. The Minimal Deliberation approach proposed by Chien[6] combines reaction-based control with a classical planning system to create plans for dealing with situations where reaction rules fail. Both of these systems require extensive domain

knowledge in the non-reactive planning systems in order to deal with limitations in reactive control. Unlike these systems, our approach does not fall back on slow non-reactive techniques for improving reactive control.

The behavior based learning system proposed by Maes is a distributed algorithm in which the behaviors learn when to become active based on feedback during execution[9]. One difference between this and our work is that our system learns varying levels of behavior (schema) activation rather than merely active versus inactive.

3. Schema-Based Navigation

Schema-based robotic control is one form of reactive control system. It forms the reactive basis of AuRA, The Autonomous Robot Architecture[2], a framework for providing reactive control for an autonomous mobile system. Each motor schema represents one fundamental motor behavior currently available to the robot. The action to be performed is the result of collectively *summing* the contributions of all currently active schemas rather than arbitrating between them. The motor schemas are closely linked to perceptual schemas which provide the environmental stimuli for activation.

Once a given motor schema is activated, its impact on the overall motion is dependent on that schema's gain value. If a schema currently has a very high gain value, that schema's behavior will likely dominate the command passed to the robot drive system. If a single schema gain value remains consistently high, that schema will dominate the overall system behavior[1].

While many possible schemas have been defined, the motor schemas used in this particular research are:

- **Move-to-goal**: Attract to goal with variable gain. Set high when heading for a particular goal.

  $V_{magnitude}$ = adjustable gain value
  $V_{direction}$ = in direction towards perceived goal

- **Avoid-Obstacle**: Repel from object with variable gain and sphere of influence. Used for collision avoidance.

  $O_{magnitude} = 0$ for $d > S$

  $\frac{S-d}{S-R} * G$ for $R < d \leq S$

  ∞ for $d \leq R$

  where:

  S = Adjustable Sphere of Influence (radial extent of force from the center of the obstacle)

  R = Radius of obstacle

  G = Adjustable Gain

  d = Distance of robot to center of obstacle

  $O_{direction}$ = along a line from robot to center of obstacle moving away from obstacle

- **Noise**: Random wander with variable gain and persistence. Used to overcome local maxima, minima, cycles, and for exploration.

  $N_{magnitude}$ = Adjustable gain value

  $N_{direction}$ = Random direction that persists for $N_{persistence}$ steps ($N_{persistence}$ is adjustable)

To optimize system performance it is necessary to to determine what gain values should be used to best accomplish a specific task in a given environment. For instance, an exploration behavior can be observed by providing a relatively high gain and persistence to the **noise** schema with an accompanying **avoid-obstacle** schema[1]. The task of determining appropriate *a priori* gain values is non-trivial in highly cluttered environments.

For a given environment, this gain determination process involves empirical evaluation of the system's performance. The process is repeated until further changes result in no visible improvement. When structural environmental knowledge is available, this task becomes simpler[2], but for purely reactive systems with no knowledge of the world, highly complex environments can produce difficulty in reaching near optimal solutions.

In any case, it is important to note that once this "best set" of gain values is established for a given world, it will likely be less efficient for navigation in a different environment. Figures 1 and 2 illustrate a scenario where this is true. Figure 1 shows a

Figure 1: Box Canyon Sample Runs
a) With sphere of influence = 10 b) With sphere of influence = 35

box canyon where a high **avoid-obstacle** sphere of influence is necessary to produce a successful path. Figure 2 shows a slightly different picture where a high sphere of influence results in a less than optimal path. High levels of **noise** were used in these sample runs producing the relatively jagged paths when the robot is located in free space. This noise is necessary to free the robot from local minima that are produced by obstacle formations in highly cluttered environments. Later in this paper, it can be seen that our on-line learning system smooths out this **noise** when it is not required.

The result of this research effort in adaptive reactive control is a system that automatically adjusts the appropriate schema gain values to match the current task and environment.

4. Control Strategy for On-Line Performance Enhancement

This section outlines a reactive robot control system incorporating machine learning techniques to improve goal finding performance. We utilize an adaptive real-time learning system that *does not require pre-training* but continuously *learns by doing.* A *critical* aspect of a learning system for robotic control is that the learning process must not interfere with the system's reactive response. A robot operating in a potentially hostile environment cannot stop to update a model when new information is found. This aspect of robotic systems is a central motivation for the current work.

Figure 2: No Box Canyon Sample Runs
a) With sphere of influence = 10 b) With sphere of influence = 35

Figure 3 shows a conceptual model of the control regime. As in our previous work[1], the basic control system is a collection of motor schemas that cooperate/compete to control the robot's behavior. In addition, we have added a component termed the ADJUSTER which provides the learning aspect of the system. The ADJUSTER calculates status values in order to track current progress and then adjusts the active schema gain values based on the newly determined status. The schema adjustments are made according to a set of rules which match current progress status values to plausible adjustment operations. The values currently adjusted are **noise** persistence ($N_{persistence}$), **avoid-obstacle** sphere of influence (S), and the three motor schema gain values ($N_{magnitude}, V_{magnitude}, G$). An example of the use of these adjustment rules is to allow the robot to increase speed by increasing the **move-to-goal** gain when no obstacles are near and reasonable progress is being made. The rules then reverse this trend and increase the **noise** and **avoid-obstacle** gains when obstacles are encountered.

4.1 Status Monitoring

A central aspect of the ADJUSTER system involves how the system monitors the robot's performance and determines when to apply the specific rules. The schema-based control system does not contain a world model with which to compare the currently perceived environment in order to determine the status of the current mission. Instead, the ADJUSTER relies on recent experience and a set of heuristics for identifying when

Figure 3: Control Strategy

good progress is being made.

We use the term **step** to indicate the action taken during a single perception-action cycle. The number of steps taken to complete a task is used as a measure of elapsed time. The robot's **step size** refers to the distance traveled in one step. This is a normalized value calculated from the relative contributions of all active schemas. A combined schema value greater than or equal to the maximum step size results in a maximum step. On a real system this is constrained by the maximum velocity of the robot. (In our simulation environment described in section 5 we use a maximum step size of 1.) The system keeps a history of the last H_{steps} steps which it uses to determine mission status. Progress is measured using the motion value M which is obtained from the average step size over this history. If M drops below a given threshold, T_{motion}, we consider the robot to be stationary for all practical purposes. This is usually the result of a local minima which can be overcome by increasing the **noise** gain and persistence.

Closely related to the average step size value is P, the measure of the average progress toward the goal. This is determined by the ratio of distance traveled over the change in distance to the goal. A low value here indicates that the robot is moving but not heading toward the goal. It is important to note that, at times, this may be a necessary situation in order to navigate around an obstacle. The challenging aspect is to correctly interpret a low or negative value as either a necessary detour or as an indication that the robot is on a completely ineffective path.

In order to determine the likely meaning of the average motion toward goal value, the ADJUSTER uses O_{count}, the average number of obstacles sensed. If there is a low

progress toward the goal and O_{count} is high the ADJUSTER assumes this is a cluttered environment.

These status measures are evaluated over a variable size history of steps, H_{steps}. The history size controls the reactivity of the ADJUSTER to changes in the mission status. In general, a shorter step history leads to more abrupt changes in the rules applied by the ADJUSTER. For most of our simulation work we used a history size of 10 steps.

4.2 Adjustment Rules

The ADJUSTER works by periodically applying slight adjustments to the control values based on the current status of the robot. These adjustments are made after H_{steps} steps based on the status values M, P, and O_{count}. We have identified four basic situations that require different adjustment parameters. These situations are:

1. **No-Movement**: The robot's average step size has dropped below a certain threshold.

   $M < T_{movement}$

   where:

   M = Average Step Size over H_{steps}

   $T_{movement}$ = Movement Threshold

2. **Movement-Toward-Goal**: The robot's step size and rate of approach to the goal are both above a threshold.

   $M > T_{movement}$

   $P = \frac{H_{distance}}{H_{goal}} > T_{progress}$

   where:

   $H_{distance}$ = Distance Traveled over H_{steps}

   H_{goal} = Decrease in Distance To Goal over H_{steps}

   $T_{progress}$ = Progress Threshold

3. **No-Progress-With-Obstacles**: The robot is moving but not toward the goal and there are several obstacles within a sensible distance.

   $M > T_{movement}$

   $P = \frac{H_{distance}}{H_{goal}} < T_{progress}$

   $O_{count} > T_{obstacles}$

   where:

   O_{count} = Average Number of Sensible Obstacles over H_{steps}

   $T_{obstacles}$ = Obstacle Count Threshold

4. **No-Progress-No-Obstacles**: The robot is moving but not toward the goal and there are no obstacles within a sensible distance.

   $M > T_{movement}$

   $P = \frac{H_{distance}}{H_{goal}} < T_{progress}$

   $O_{count} < T_{obstacles}$

The actual definitions of these situations have evolved over the span of this work. In the initial ruleset the **Movement-Toward-Goal** rule was divided into separate rules for situations with and without obstacles. In most cases these rules were redundant and they were collapsed into one. Also, there was initially no distinction made between movement and movement toward the goal. This lack of distinction led to situations where the robot could get stuck in local minima behind obstacles.

Associated with each rule is an adjustment value specifying the amount and direction which each of the five control values should be adjusted when the rule is invoked. For each schema value there is also a range specifying the minimum and maximum allowable values. These ranges prevent the ADJUSTER from continuing to make changes to a value beyond a reasonable limit. The adjustment values to be used for a mission are specified in a case[8] and stored in a datafile. When a particular mission is initiated, the appropriate case is retrieved and the rule adjustment values are established. Our experiments with the ADJUSTER involved changing these rule adjustment values and comparing the results.

	Noise		Goal	Obstacle	
Rule	$N_{persistence}$	$N_{magnitude}$	$V_{magnitude}$	G	S
No-Movement	+1.00	+0.10	−0.10	−0.10	−0.50
Movement-To-Goal	−1.00	−0.05	+0.05	−0.01	−0.50
No-Progress-With-Obstacles	+1.00	+0.05	−0.05	+0.01	+0.50
No-Progress-No-Obstacles	−1.00	−0.05	+0.05	+0.01	+0.50

Table 1 - Sample ADJUSTER Case

Table 1 shows the adjustment values found in a sample case. Each value indicates the amount of change for the given control parameter when the associated rule is triggered. The first rule in this case is the **No-Movement** rule which is invoked when the robot stops moving. In order to recover from this situation the system increases the importance of the **noise** schema. This is done by increasing $N_{persistence}$ and $N_{magnitude}$, decreasing $V_{magnitude}$, and decreasing obstacle influence by lowering G, gain, and S, sphere of influence.

The tendency toward building momentum is indicated in the **Movement-Toward-Goal** rule. This rule enhances the importance of the **move-to-goal** schema by a decrease in $N_{persistence}$ and $N_{magnitude}$, an increase in $V_{magnitude}$, and a decrease in obstacle influence by lowering G and S. When the system is making progress toward the goal in an uncluttered area, applying these adjustments will improve that progress.

If the robot moves into a more cluttered environment resulting in a decrease in progress toward the goal, the ADJUSTER begins to consider the other three adjustment rules. The first rule it will likely use is the **No-Progress-With-Obstacles** rule. In this particular case the adjustment values applied are the exact opposite of the adjustments applied by the **Movement-Toward-Goal** rule. This increases both the importance of obstacles and the application of noise. The assumption made in applying this rule is that the best way to navigate a cluttered environment is to steer as far away from obstacles as possible. In many sample cases this proved to be the most effective choice. Later we will discuss some alternatives to this approach.

After the ADJUSTER applies the above **No-Progress-With-Obstacles** rule the robot will usually move around a large collection of obstacles. If several applications of this adjustment rule were required before successfully circumventing an obstacle it is possible the $V_{magnitude}$ will have been decreased to a point where no progress will be made even though there are no obstacles sensed. In this situation the **noise** schema begins to dominate the control system. Applying the **No-Progress-No-Obstacles** rule will allow the system to recover from this situation.

5. Simulation Environment

In order to facilitate our work in adaptive control systems for autonomous navigation, we have developed a simulation environment that includes both a graphical interface and batch mode. This simulation system allows the investigator to visually evaluate the progress of a simulated robot while it runs through a predefined world. It also displays a history of values as they are adjusted by the system to facilitate the identification of successful adjustment rules.

The graphical interface was developed using the X window system on a Sun workstation. The simulation window displays the current obstacles as circles, each with varying radius. As the robot navigates this world, a line is drawn indicating the robot's progress from start to goal. At the top of the window is a set of numbers displaying the current control values. These values are updated each time the ADJUSTER is called. This display also indicates the number of steps, total distance traveled, distance to goal, and number of obstacle contacts. Below the numerical display is a set of five line graphs that provide a history of the control values as they are adjusted throughout the run. This window format is used in Figures 4-7.

Within each obstacle is displayed a number representing the current impact of that obstacle on the robot's move calculation. These values are changed after every step as the robot's distance from the obstacle and the obstacle G and S values change. As a simulation runs, the mouse can be used to pause and run in single step mode. This allows the user to check the impact of specific obstacles for each step and determine which adjustment rules are being used in a given situation.

The obstacle world and initial parameters are input to the simulator from a text file. This file allows the user to specify obstacle courses and set the initial control values and adjustment rule parameters. There is a random world generator that creates a random start and destination goal and fills in the path between with a random distribution of obstacles. By specifying a percentage of clutter, worlds of varying difficulty can be created.

6. Results

The simulation environment was used to run large numbers of simulations both interactively and in batch mode. This section outlines some of the observations made after analyzing these runs.

6.1 Overall Improvement

One of the first goals of our simulation work was to verify the improvement of the system with the ADJUSTER over the regular schema-based control system in a wide variety of environments. To do this the simulator's random world generator was used to create ten different worlds at each of four different obstacle densities. The simulation results for each of these worlds were recorded both with and without the ADJUSTER enabled. Table 2 contains the results of the runs without the ADJUSTER averaged over the ten different worlds of each density level. Table 3 contains the results with the ADJUSTER enabled. In these tables, the number of steps is an indication of the average time taken to complete each mission. Total distance indicates the total length of the paths taken. Average step size is the total distance divided by the number of steps and is a good comparison to use between worlds with varying initial goal distances. The contacts value indicates the number of times the robot *virtually* contacts an obstacle (as a configuration space methodology is used and a safety margin attached to the obstacle, these contacts are indicators that the robot has moved unacceptably close to the object but has not necessarily physically contacted it). These contacts force obstacle avoidance to be the primary concern by resulting in immediately high gains for the obstacle in question thus preventing actual physical collisions.

Course	% Complete	Steps	Total Distance	Avg Step Size	Contacts
No Obstacles	100	49.30	42.09	0.86	0.00
25% Obstacles	100	192.70	109.74	0.66	1.40
50% Obstacles	90	868.70	431.87	0.54	24.60
75% Obstacles	80	1134.75	545.94	0.61	32.00

Table 2 - Results Without ADJUSTER Over 10 Randomly Generated Worlds

Course	% Complete	Steps	Total Distance	Avg Step Size	Contacts
No Obstacles	100	44.20	40.07	0.91	0.00
25% Obstacles	100	54.20	45.21	0.81	0.00
50% Obstacles	90	95.67	61.41	0.69	1.33
75% Obstacles	90	210.43	133.06	0.68	21.57

Table 3 - Results With ADJUSTER Over 10 Randomly Generated Worlds Using Squeezing Strategy

For all four obstacle density levels, the ADJUSTER makes a significant reduction in the total distance traveled (5%-75% less than non-adjusted values). The average step size increases correspondingly, in essence allowing the robot to make longer strides. For the first density level there are no obstacles. At this level the only two contributing gain values are **noise** and **move-to-goal**. In the unadjusted case, the $N_{magnitude}$ is set to a constant (0.5) value and makes a small contribution to the run. The resulting robot

path is close to optimal but not straight. In the runs with adjustment, the ADJUSTER applies the **Movement-Toward-Goal** rule which decreases the $N_{magnitude}$ value so that the robot's path improves toward a straight line. This is reflected by the decrease in total distance traveled as well as the increase in step size.

The second density level is 25% cluttered. This means that 25% of the area between the start and goal is covered by obstacles. In this case the time and total distance values improve substantially in the runs with the ADJUSTER enabled. This improvement is a reflection of the robot's improved ability to react to local minima situations and move out from behind obstacles encountered along the way. These runs also showed speedups when the robot enters open space and the $N_{magnitude}$ value decreases.

The 50% and 75% levels also show substantial improvement in both the number of steps taken and total distance traveled. They also show improvement by decreasing the number of contacts as well as increasing the percentage of missions completed. Figures 4 and 5a show this improvement with an example of the 50% clutter. As the worlds become more cluttered, there is a greater improvement when the ADJUSTER is used.

6.2 Modifying ADJUSTER Parameters

Another goal of our simulations is to determine the most successful adjustment rule parameters. One of the areas explored was what the robot should do when it is not making progress with obstacles present. One possibility is to assume that there exists a path between the obstacles and reduce the **avoid-obstacle** G and S in order to pass between them. We refer to this approach as a *squeezing strategy*. This is the strategy used on the previous comparison to the case with no adjustment. (Table 3) The other alternative, which we term a *ballooning strategy*, is to assume that the current situation is a box canyon and increase G and S in order to quickly move around the obstacle collection.

Course	% Complete	Steps	Total Distance	Avg Step Size	Contacts
25% Obstacles	100	68.10	53.72	0.82	0.20
50% Obstacles	70	112.33	79.54	0.67	1.00
75% Obstacles	70	905.29	677.03	0.72	8.57

Table 4 - Results With ADJUSTER Over 10 Randomly Generated Worlds Using Ballooning Strategy

Table 4 shows the data from the simulation runs where the ballooning strategy was used and Figure 5b shows a sample of one of these runs. When we compare these results to those in Table 3 we see that the robot generally takes more time and travels a longer path. It also successfully completes fewer paths. This is due to the fact that in

Figure 4: 50% Obstacle Run Without ADJUSTER

Figure 5: 50% Obstacle Run With ADJUSTER
a) Squeezing Strategy b) Ballooning Strategy

the more cluttered worlds the robot may be surrounded by obstacles so that increasing S will prevent making any progress at all. One positive result of this adjustment rule is a decrease in the number of obstacles virtually contacted. Figure 6 shows an example comparing three different runs in a world with 75% obstacle clutter.

6.3 The Box Canyon

Another goal of this work has been to develop a reactive control system that would successfully navigate through an environment where *box canyons* or dead end paths are present. Table 5 summarizes the results of our experiments with box canyon environments. Each value is the average of ten runs through the same box canyon environment. The system without the ADJUSTER enabled does not successfully navigate to the goal. (ie, It covers an infinite distance and takes infinite number of steps.) In this situation the robot becomes caught in a local minima within the canyon where the **avoid-obstacle** schemas for the surrounding obstacles cause a contribution equal to the **move-to-goal** schema. The distance out of the canyon, (the depth of the local

Figure 6: 75% Obstacle Runs
a) Without ADJUSTER b) With ADJUSTER - Squeezing Strategy
c) With ADJUSTER - Ballooning Strategy

Figure 7: Box Canyon Run With ADJUSTER -
a) Max Noise Persistence 5 b) Max Noise Persistence 8

minima), is farther than a normal **noise** gain value will overcome.

Course	% Complete	Steps	Total Distance	Avg Step Size
No ADJUSTER	0	∞	∞	
Max $N_{persistence} = 5$	50	2273.80	1859.69	0.81
Max $N_{persistence} = 8$	100	1600.40	1205.16	0.75

Table 5 - Box Canyon Navigation

The first successful navigation runs in this environment use the same ADJUSTER case rules as the ballooning strategy case discussed above. This is appropriate since there is no opening between the obstacles which would allow the robot to pass through toward the goal. In these runs, the robot successfully navigated to the goal in 50 percent of the cases. This is a significant improvement over the runs without the ADJUSTER but still leaves room for further enhancement. Figure 7a shows a sample run for this situation. Note the top line graph indicating the fluctuation in $N_{persistence}$ as the robot backs up and then begins to move toward the goal again without actually moving around the obstacles.

After analyzing the paths taken in these runs we determined that the system was succeeding when the **noise** schema would generate several successive values in the direction heading out of the canyon. We hypothesized that increasing the maximum allowable $N_{persistence}$ from the current value of 5 to a higher value might improve the performance. The third result in Table 5 shows the average of 10 runs with a maximum $N_{persistence}$ value of 8. Each of these runs were successful and the resulting distance traveled decrease by just over one third. By allowing $N_{persistence}$ to increase to a larger value we allowed the robot to be influenced by the **noise** schema long enough to *find its way* around the end of the canyon wall. Figure 7b shows a sample run of the simulator through a box canyon with the maximum $N_{persistence}$ set at 8.

This increased allowable persistence is not used without some cost. As is visible in Figure 7b, it takes more time for the robot to regain forward momentum after rounding the wall and entering uncluttered space with a clear path to the goal. The decreasingly jagged path to the goal is a visible indication of the systems increasing momentum as it applies the **Movement-Toward-Goal** rule and decreases the **noise** contribution. This situation only occurs in open space navigation. We expect the benefit of the higher $N_{persistence}$ value in cluttered worlds to outweigh the cost of some efficiency in uncluttered space.

7. Conclusions and Future Work

We have demonstrated a reactive robotic system that incorporates ideas from machine learning to address a particular issue in schema-based robotic control systems. The goal of this learning system is to give the autonomous robot the ability to adjust the schema control parameters in an unstructured dynamic environment. The results of a successful implementation that learns to navigate out of a box canyon have been presented. This system never resorts to a high level planner, but instead, it learns continuously by adjusting gains based on the progress made so far. The system is successful because it is able to improve its performance in reaching a goal in a previously unfamiliar and dynamic world.

Other related areas we plan on continuing to explore include:

- Rule Persistence - We intend to explore the feasibility of adding a persistence variable for the application of the various rules. One case where this might be useful is in combining the two strategies of ballooning or squeezing. The ADJUSTER would start out by applying the statistically more successful squeezing option. If a certain persistence time has passed and the progress has not improved, the ballooning rule could be applied. This could potentially allow the

robot to navigate out of a deep box canyon that initially looks like an open tunnel but actually turns out to be closed.

- Case Learning - We have already begun exploring the issues involved in applying a meta-level learning aspect to the system where the system learns the ADJUSTER cases through experience. Information would be given to the system regarding the mission expectations including: expected environmental clutter, likelihood of box canyons, and threat of obstacles. These expectations would form the indices for selecting an appropriate ADJUSTER case. After a mission is completed, the system would then evaluate the effectiveness of the selected case and determine its value for future missions.

- Implementation on our mobile robot - The ultimate goal of this project is to develop a model that can be used on a working vehicle. After successfully developing a learning control architecture through simulation studies, we plan to test the system on George, our Denning Mobile Robot. This implementation will provide insight into the practicality of this architecture in terms of dealing with real-world issues like imperfect sensor information.

8. Acknowledgments

We are grateful for the support of the Georgia Tech Artificial Intelligence Group and the assistance of Robin Murphy.

REFERENCES

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

[2] Arkin, R.C., "Integrating behavioral, perceptual, and world knowledge in reactive navigation.", *Robotics and Autonomous Systems*, 6:105-122, 1990.

[3] 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*, Vol. 20, No. 6, Nov/Dec 1990, pp. 1245-1257.

[4] Bennett, S.W., "Reducing Real-world Failures of Approximate Explanation-Based Rules", *Proceedings of the Seventh International Conference on Machine Learning*, June 1990, pp. 226-234.

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

[6] Chien, S.A., M.T. Gervasio and G.F. DeJong, "On Becoming Decreasingly Reactive: Learning to Deliberate Minimally", *Proceedings of the Eighth International Workshop on Machine Learning*, June 1991, pp. 288-292.

[7] Christiansen, A.D., M.T. Mason and T.M. Mitchell, "Learning Reliable Strategies without Initial Physical Models", *IEEE Conf. on Robotics and Auto.*, pp. 1224-1230, 1990.

[8] Kolodner, J.L., *An Introduction to Case-Based Reasoning*, Technical Report No. GIT-ICS-90/19. College of Computing. Georgia Institute of Technology. Atlanta, GA., 1990.

[9] Maes, P., R.A. Brooks "Learning To Coordinate Behaviors", *Proceedings of the Eighth National Conference on Artificial Intelligence*, Boston, MA, Aug 1990, pp. 796-802.

[10] Mitchell, T.M. "Becoming Increasingly Reactive", *Proceedings of the Eighth National Conference on Artificial Intelligence*, Boston, MA, Aug 1990, pp. 1051-1058.

[11] Payton, D., "An Architecture for Reflexive Autonomous Vehicle Control", *IEEE Conf. on Robotics and Auto.*, pp. 1838-1845, 1986.

[12] Zelinsky, A., *A Mobile Robot Environmental Learning Algorithm*, Department of Computing Science, University of Wollongong, April, 1988.