Spatio-Temporal Case-Based Reasoning for Behavioral Selection*

Maxim Likhachev and **Ronald C. Arkin**
Mobile Robot Laboratory
College of Computing, Georgia Institute of Technology
mlikhach@cc.gatech.edu, arkin@cc.gatech.edu

Abstract

This paper presents the application of a Case-Based Reasoning *approach to the selection and modification of behavioral assemblage parameters. The goal of this research is to achieve an optimal parameterization of robotic behaviors in run-time. This increases robot performance and makes a manual configuration of parameters unnecessary. The case-based reasoning module selects a set of parameters for an active behavioral assemblage in real-time. This set of parameters fits the environment better than hand-coded ones, and its performance is monitored providing feedback for a possible reselection of the parameters. This paper places a significant emphasis on the technical details of the case-based reasoning module and how it is integrated within a schema-based reactive navigation system. The paper also presents the results and evaluation of the system in both in simulation and real world robotic experiments.*

Index terms: *Case-Based Reasoning, Behavior-Based Robotics, Reactive Robotics.*

I. INTRODUCTION

Reactive control for robotics is known to provide good performance in unknown or dynamic environments. Such environments provide very little a priori knowledge and often no time for deliberation. A reactive system provides a tight coupling of perceptual data to an action, and a response to a current stimulus is produced without any deliberation. At any point of time, based on incoming sensory data, a robot selects a subset of behaviors (behavioral assemblage) from the set of predefined behaviors and then executes them. One of the problems, however, of this approach is that as an environment gradually changes, the selected behaviors should also be adjusted correspondingly or re-selected to represent a behavioral assemblage that better fits a new situation. However, increasing the parameter space of the predefined behaviors to cover all the possible types of environments in the world would be impractical as the types of environments represent a continuous space and would require an extremely large set to produce a good approximation to the optimal behavior space; building each optimal set of parameters to fit a particular type of environment is very tedious work requiring a significant number of simulations and real robot experiments. Second, it cannot be expected to know in advance when and which types of environments a robot will encounter during its mission. Finally, it is desirable to make the process of mission specification as user-friendly and quick as possible by not requiring a user to adjust the parameters of behaviors. This paper presents a solution to this problem based on the incorporation of case-based reasoning into the behavior selection process. This permits an automatic selection of optimal parameters in run-time while the mission specification process no longer requires manual configuration of behavioral parameters.

As robot executes its mission a case-based reasoning (CBR) unit controls switching between different sets of parameters in response to a change in the environment type. Each such set of parameters constitutes a case in the CBR system. The adaptation step in the case-based reasoning procedure fine-tunes the parameters to a specific type of environment allowing the library of cases to be small. The overall hierarchy is shown in Figure 1 below.

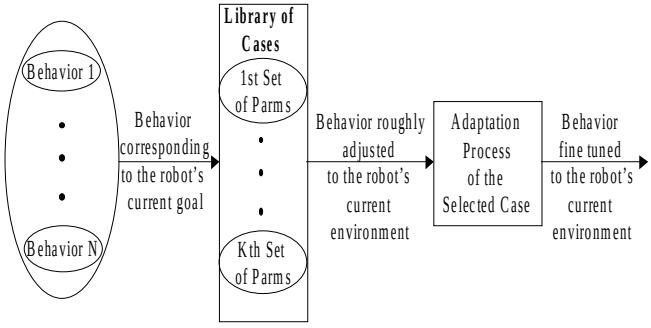

Figure 1. Behavioral Selection Process with case-based reasoning unit incorporated.

A case-based reasoning methodology is not new to the field of robotics. It was successfully used to help in solving such problems as path planning based on past routes, high-level action-selection based on environment similarities, place learning, and acceleration of complex problem solving based on past problem solutions [3, 4, 5, 6, 7, 8]. Previous work has also been performed on the incorporation of case-based reasoning in the selection of behavior parameters by our group [1, 2] on which this

* This research is supported by DARPA/U.S. Army SMDC contract #DASG60-99-C-0081. Approved for Public Release; distribution unlimited.

present research is partially based and a few others [e.g., 13]. The approach described in this paper, however, differs significantly from the previous algorithms by introducing: a novel feature identification mechanism that produces spatial and temporal vectors describing the current environment; a notion of *traversability vectors* that measure the degree of traversability around a robot in configurable number of directions; a randomization in the case selection process to allow for the exploration of cases; and a case switching decision tree to adaptively control case switching based on a case performance. This novel methodology results in very robust performance of the robot while allowing for easy input and output vector space representation, straightforward extensions and modifications, and simple control of computational complexity depending on the available computational resources and precision of sensor data. The work presented in this paper also extends the previous work [1, 2] by incorporating the case-based reasoning within a hybrid robot architecture and evaluating the performance on both real and simulated robots.

II. METHODOLOGY

A. Framework

The framework chosen for the integration of the case-based reasoning for behavioral selection is the *MissionLab* system [9], which is a version of AuRA (Autonomous Robot Architecture) [10]. The overall hybrid architecture consists of a schema-based reactive system coupled with a high-level deliberative planning system. The reactive component consists of primitive behaviors called motor schemas [11] grouped into sets called behavioral assemblages. Each individual primitive behavior is driven by its perceptual input (perceptual schema) producing its own motor response. The vectorial responses from each of the active schemas are added together resulting in an overall behavior output. The weighted sum of the vectors, after normalization, defines the final vector that is sent to the motor actuators. Hence, each motor schema affects the overall behavior of the robot.

Within *MissionLab*, a finite state automaton defines the high-level plan of a robot's mission. Each state in the plan is one of the predefined behavioral assemblages chosen to achieve the goal of a robot at this state. The transitions between states are triggered by perceptual inputs called triggers.

B. Integration of CBR within MissionLab

Every behavioral assemblage (a state in a high-level plan) is controlled by a set of parameters. Normally, these parameters would be carefully chosen by a user to correspond to the task-environment the robot is expected to inhabit. If optimal behavior for the robot is desired, states could be split into multiple states with the same behavioral assemblage but with different sets of parameters where each is adjusted to particular environmental characteristics. This method would also require designing special perceptual triggers to detect the changes in environment type. This complexity is avoided by employing the case-based reasoning unit that, for a currently chosen behavioral assemblage, selects in real-time the set of parameters that is best suited to the current environment. As the type of environment might change unexpectedly, the CBR unit continually monitors and re-selects and re-adapts the assemblage parameters as necessary.

The diagram of how the CBR unit is integrated within *MissionLab* appears in figure 2. The sensor readings enter into both the high-level FSA-based planner and the CBR unit. Based on the perceptual input, the same or a new state is selected. The chosen state defines a behavioral assemblage that is then passed into the behavioral control unit. The chosen state identifier is also passed into the CBR unit along with relevant information about the current robot's goal (e.g., goal position). If the CBR unit supports the current state, then based on the perceptual input, goal information and the state, a set of parameters for the behavioral assemblage is selected from the case library and adapted to better fit the environment. These parameters are passed into the behavioral control unit, which applies them to the current behavioral assemblage. After evaluating this assemblage, the motor vector is produced and supplied to the actuators for execution. If the chosen state, however, is not supported by the CBR unit, then the CBR unit passes a special flag to the behavioral control unit, and the behavioral assemblage is used with its default parameter values as defined in the finite state machine.

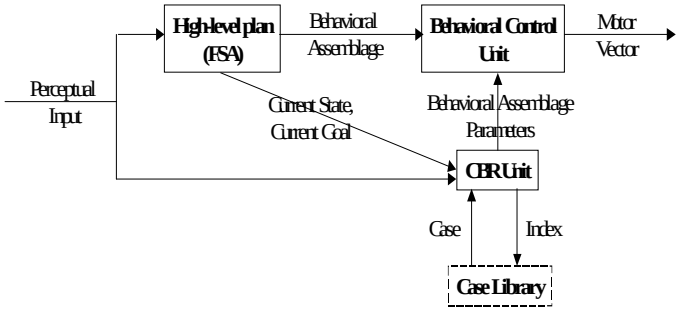

Figure 2. Integration of the case-base reasoning unit within the AuRA architecture.

Currently, the CBR unit supports navigational states of type **GOTO**. The behavioral assemblage that corresponds to such a state includes the following motor schemas: The **MoveToGoal** schema produces a vector directed toward a goal location from the current robot's position. The magnitude of this vector called *MoveToGoal_Gain* is an adjustable parameter for this schema. The **Wander** schema generates a random direction vector, adding an exploration component to the robot's behavior. This schema has two parameters: random vector magnitude called *Noise_Gain;* and *Noise_Persistence*, which control the rate of directional switching of the vector. The **AvoidObstacles** schema produces a vector that results from repelling forces from each of the obstacles within some distance from the robot. Each repulsive force is a vector with a direction from the detected obstacle toward the robot and the magnitude given in formula (1).

$$O_{mag} = \begin{cases} 0, & \text{for } d \geq S \\ \dfrac{S-(d-M)}{S}, & \text{for } M \leq d < S \\ \infty, & \text{for } d < M \end{cases} \qquad (1)$$

where d is the distance at which the obstacle is detected by sensors, M is the safety margin, and S is the sphere of influence that controls the distance beyond which obstacles do not affect the robot behavior. The magnitude varies linearly from 0 to 1, when a robot is within the sphere of influence from an obstacle. As the robot comes inside the safety margin of the obstacle, the magnitude becomes very large. All the vectors are summed to produce one final vector as an output from the AvoidObstacles schema. The vector is not normalized in order to preserve the effect of infinite repulsion when the robot is too close to any obstacle. The parameters controlling the schema are the sphere of influence called *Obstacle_Sphere* and the gain called *Obstacle_Gain* that is applied via multiplication to the output vector. The final schema in the GOTO behavioral assemblage is the **BiasMove** schema, which produces a vector in a certain direction in order to bias the motion behavior of the robot. The direction of the vector, *Bias_Vector_X* and *Bias_Vector_Y*, and its magnitude, *Bias_Vector_Gain*, are the parameters that control this schema.

Thus, the output space of the CBR unit for the GOTO state is defined by the following vector:

<Noise_Gain, Noise_Persistence, Obstacle_Sphere, Obstacle_Gain, MoveToGoal_Gain, Bias_Vector_Gain, Bias_Vector_X, Bias_Vector_Y >

A case in a library is a set of values for these parameters.

C. Case-Based Reasoning Unit Overview

The overall structure of the CBR unit is similar to a traditional non-learning case-based reasoning system [12] (figure 3). The sensor data and goal information is supplied to the Feature Identification sub-module of the CBR unit. This sub-module computes a spatial features vector representing the relevant spatial characteristics of the environment and a temporal features vector representing relevant temporal characteristics. Both vectors are passed forward for a best matching case selection.

During the first stage of case selection, all the cases from the library are searched, and the distances between their spatial feature vectors and the environmental spatial feature vector are computed. These distances define spatial similarities of cases with the environment. The case with the highest spatial similarity is *the best spatially matching case*. However, all the cases with a spatial similarity within some delta from the similarity of the best spatially matching case are selected for the next stage selection process. These cases are called *spatially matching* cases. At the second stage of selection all the spatially matching cases are searched, and the distances between their temporal feature vectors and the environmental temporal feature vector are computed. These distances define temporal similarities of cases with the environment. The case with the highest temporal similarity is *the best temporally matching case.* And again, all the cases with a temporal similarity within some delta from the similarity of the best temporally matching case are selected for the next stage selection process. These cases are *spatially and temporally matching* cases and are all the cases with close spatial and temporal similarity to the current environment. This set usually consists of only a few cases and is often just one case. The set, however, can never be empty as the most similar to the environment case is always selected into it, independently of how dissimilar the case might be.

The last selection stage is a uniformly random selection from the set of spatially and temporally matching cases. The idea is that these cases are all close enough to the current environment. Their output parameter vectors, however, might be very different. A specific pair of temporal and spatial feature vectors does not necessarily map onto an optimal solution due to possible aliasing. As a result, all the cases sufficiently similar to the current environment deserve a chance to be tried.

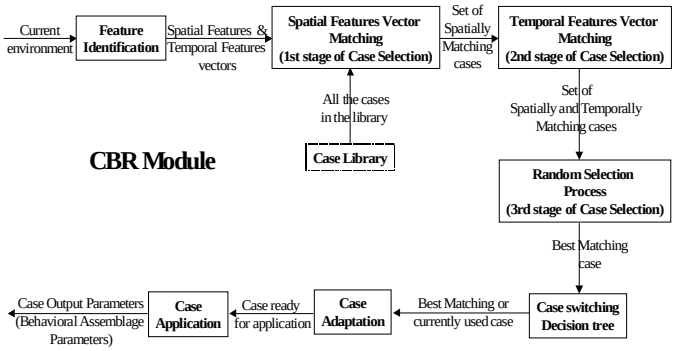

Figure 3. High-level structure of the CBR Module.

The case switching decision tree is then used to decide whether the currently applied case should still be applied or should be switched to the case selected as the best matching one. This protects against thrashing and overuse of cases. If a new case is to be applied, then it goes through the case adaptation and application steps. At the adaptation step, a case is fine-tuned by slightly readjusting the behavioral assemblage parameters that the case contains to better fit the current environment. At the application step these parameters are passed on to the behavioral control module outside of the CBR unit.

D. Case-Based Reasoning Unit: Technical Details

1) Feature Identification Step

In this step spatial and temporal feature vectors are produced based on current environment data. This data includes sensor readings and goal position. The sensor data are distances to obstacles along rays shot from each of the sensors on the robot (e.g., as produced by sonar or laser sensors).

The spatial feature vector has two elements in it: a distance from the robot to the goal, *D*, which is a scalar and a sub-vector which represents an approximation of obstacle density function around the robot and is computed as follows (Fig. 4). The space around the robot is divided into K angular regions. The regions are always taken in such a

way that the bisector of the 0th region is directed toward the goal of the robot. Within each region the cluster of obstacles that obstructs the region most of all is found. An obstacle density function approximation vector is then represented by K pairs, $<\sigma, r>$, where σ is the degree of obstruction of a region by the most obstructing cluster in this region, and r is the distance to this cluster.

Figure 4 demonstrates an example computation of the obstacle density. There are 12 sensors in this example. They are evenly spaced around the robot, which is located in the center of the large circle. The large circle is a clipping circle where all the obstacles detected beyond that circle are ignored in the computation of the density function. The circled obstacles within each region define the most obstructing clusters within each region. Their corresponding degree of obstruction, σ, is then computed as the ratio of the angle that they obstruct within the region over the angle of the whole region. Thus, σ is equal to 1.0 for region 1 indicating that the obstacles obstruct the region completely, whereas σ is equal to 2/3 for the 0th and 3rd regions, as the obstacles leave 1/3 of the angle in the regions free for traversing. Region 2 has σ equal to 0.0 since there are no obstacles detected within the region's clipping circle. Thus, the whole region is available for traversing.

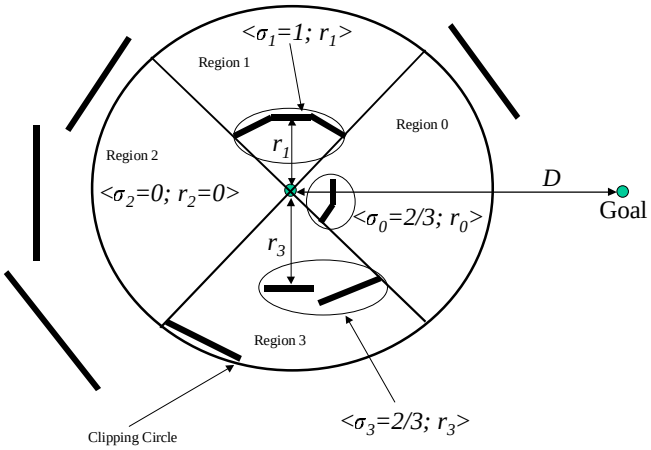

Figure 4. Computation of the spatial feature vector for K=4 (4 regions). The robot is in the center of the circle. Thick lines are obstacles as detected from 12 sensors evenly placed around the robot. The circled clusters of obstacles within each region are the most obstructing clusters.

Figure 4 also shows the distance of the robot to the goal, D, which is the first element of the spatial feature vector. Note that the number of regions, K, is determined based on the desired computational complexity and also the resolution of sensor data. If K is equal to the number of sensors on the robot, then the obstacle density function is the actual raw sensor data clipped by the clipping region. Thus, in the above example setting K to more than 4 might not bring any benefit as there are only 3 sensors per region anyway.

The temporal feature vector contains two scalar elements: short-term relative motion, R_s, and long-term relative motion, R_l. The short- and long-term relative motion measures represent short- and long-term velocities of the robot, respectively, relative to the maximum possible velocity of the robot and are computed as shown in formula (2). The same formula is used for the computation of both relative motion measures. However, the time window lengths used to compute average robot positions differ between long- and short-term relative motion computations as shown in the formula (3).

$$R_i = \frac{\left\| Pos_{i,longterm} - Pos_{i,shortterm} \right\|}{N * MaxVel}, \text{ for } i = s, l \quad (2)$$

where N is the normalization constant, $MaxVel$ is the maximum robot velocity, and $Pos_{i,longterm}$ and $Pos_{i,shortterm}$ are average positions of the robot over long- and short-term time windows, respectively, and are updated according to formula (3) every time the CBR module is called.

$$Pos_{i,j} = a_{i,j} * Pos_{i,j}^{old} + (1 - a_{i,j}) * NewPos \quad (3)$$

for $i = s, l$ and $j = shortterm, longterm$

where $NewPos$ is a new current position of the robot, and the filter coefficient, a, is dependent on whether $Pos_{s,shortterm}$, $Pos_{s,longterm}$, $Pos_{l,shortterm}$ or $Pos_{l,longterm}$ is computed. Thus, for example, $a_{s,shortterm}$ is set to a coefficient with decay time of 5 time cycles, whereas $a_{l,longterm}$ is set to a coefficient with decay time of 600 time cycles.

Formula (4) summarizes the form of the spatial and temporal vectors.

$$V_{spatial} = \begin{bmatrix} D \\ \sigma_0, r_0 \\ \vdots \\ \sigma_{K-1}, r_{K-1} \end{bmatrix} \quad V_{temporal} = \begin{bmatrix} R_s \\ R_l \end{bmatrix} \quad (4)$$

These two vectors define input features (indices) for cases and are passed in this form into the best matching case selection described next.

2) Case Selection Process

The best matching case selection is broken into three steps. At the first step, a set of spatially matching cases is found. All the cases in the library contain their own spatial and temporal feature vectors. The similarity between spatial feature vectors for a case and environment is used to assess the degree to which the case matches the environment spatially. In order for spatial feature vectors to be comparable, however, they are first transformed into *traversability vectors.* A *traversability vector, F,* eliminates actual distances and just represents the degree to which each region can be traversed. Formula (5) presents the transformation from a spatial vector into a traversability vector.

$$F = \begin{bmatrix} f_0 \\ \vdots \\ f_{k-1} \end{bmatrix}, \quad f_i = \min(1,\ 1 - \sigma_i * \frac{D_f - r_i}{D_f}), \quad (5)$$

$$D_f = \max(D_{\min},\ \min(D_{\max},\ D))$$

where D is the distance to the goal (see equation (4)), D_{min} and D_{max} are the minimum and maximum thresholds, respectively, for considering traversability in a region, and $<\sigma_i, r_i>$ are elements of $V_{spatial}$ as defined in equation (4)

The idea is that D_f represents the circle of interest for traversability computation. The goal distance, D, limits it on one hand, while D_{max} also limits it if the goal is too far away. D_{min} is just a minimum threshold to prevent zero radius circles. The traversability measure, f_i, ranges from 0 to 1. It is proportional to the degree of obstruction, σ_i, of the most obstructing cluster in the region and the distance, r_i, at which this cluster is present in the region. Thus, if a cluster of obstacles is extremely close to the robot and it obstructs the whole region, then the region's traversability measure, f, becomes 0. If, however, obstacles obstruct the region minimally, or they are all beyond the circle of interest with radius of D_f, then the traversability measure, f, approaches 1.

To avoid large changes in the traversability vector for environment, F^{env}, due to noise in sensor data, the vector is passed through the smoothing filter given in formula (6). The coefficient b is chosen such as to have a decay time on the order of 5 to 10 sensor readings.

$$f_i^{env} = b * f_i + (1-b) * f_i^{env,old} \qquad (6)$$

where f_i is computed according to formula (5) based on the spatial vector for current environment and $f_i^{env,old}$ is f_i^{env} from the previous execution of CBR module.

Now that every case in the library is represented by a traversability vector, F, and the current environment is represented by a traversability vector F^{env}, these vectors can be used to assess the spatial similarities between the cases and environment. The spatial similarity is computed as the weighted sum of squared errors between a case and environment traversability vectors. There is significantly more weight given to the regions directed more towards the goal. This assures that, for example, if a case and environment have clear-to-goal situations in the 0th region then the environment is more similar to this case than to any other case that might have other very similar regions but does not have the clear-to-goal situation in the 0th region. Formula (7) shows the computation of spatial similarity S.

$$S = 1 - \frac{\sum_{i=0}^{K-1} w_i * (f_i - f_i^{env})^2}{\sum_{i=0}^{K-1} w_i} \qquad (7)$$

where W is the vector of weights for each region, F is the traversability vector of a case, and F^{env} is the traversability vector of the current environment. Thus, the perfect match is represented by S equal to 1, and the maximum difference by S equal to 0.

After the spatially based case selection, the set of spatially matched cases contains all the cases with spatial similarity S within some delta from the spatial similarity of the best spatially matching case. The best spatial matching case is defined as the case with the highest spatially matching similarity with the current environment.

Similarly, at the second selection step the temporal similarity with the current environment is computed for all the cases in the set of spatially matched cases according to formula (8).

$$S = 1 - \frac{w_l * (R_l - R_l^{env})^2 + w_s * (R_s - R_s^{env})^2}{w_l + w_s} \qquad (8)$$

where w_l and w_s are long- and short-term relative motion measure weights, $< R_s,\ R_l >$ is a temporal vector for a case, and $< R_s^{env},\ R_l^{env} >$ is a temporal vector for the current environment. The long-term relative motion measure is given more weight indicating its greater importance in the assessment of temporal similarities.

The best temporally matching case is the case that has the highest temporal similarity with the environment. All cases with a temporal similarity within some delta from the temporal similarity of the best temporal matching case are selected from the set of spatially matched cases for the next selection stage. Thus, after the temporal-based selection process, the set of matched cases contains the cases that are both spatially and temporally similar to the environment.

Finally, at the third and the last step of the case selection process, randomness is added to the selection process. Namely, one case from the set of matched cases is selected randomly. This selected case is declared arbitrarily to be the best matching case with the current environment.

3) Case Switching Decision Tree

At this step the decision is made as to whether the best matching case or the currently applied case should be used until the next call to the CBR module. This decision is based upon a number of characteristics describing the potential capabilities of the best matching case and the current case. The decision tree is shown in figure 5.

At the root of the tree, the time the current case was applied is checked against some threshold *CaseTime* that is specific to each case in the library. If the case was applied for less time than the threshold, then the spatial similarity of the current case is checked against threshold S_{low}, and the difference between the new case's spatial similarity and the current case's spatial similarity is checked against some threshold S_{diff}. If the two conditions are satisfied, then the current case is continued to be used. The intent is that the current case should not be thrown away too soon unless the environment became significantly

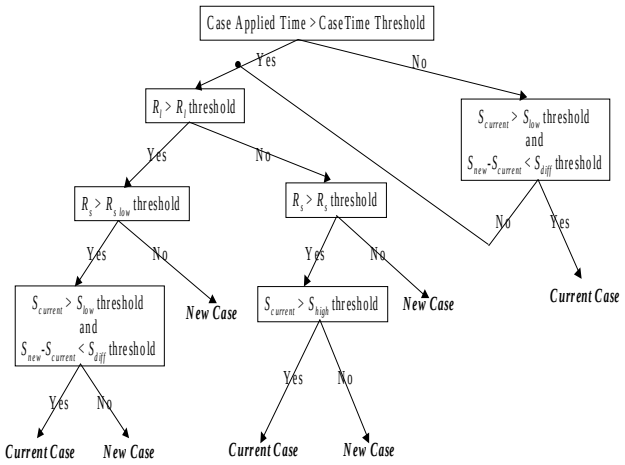

Figure 5. Case Switching Decision tree

different from what it was when the current case was initially selected. If one or both of the conditions are unsatisfied or if the case was applied for longer than the suggested threshold, the decision-making proceeds to checking the long-term relative motion measure, R_l. If it is larger than some threshold, then the case is more likely to be performing well and the short-term relative motion measure, R_s, should be compared against a low threshold – $R_{s\ low}$. If the short-term relative measure is also higher than the low threshold, it suggests that the current case performs well and it is exchanged for the new one only if its spatial similarity is very different from the environment or a much more similar case is found. Otherwise, the current case is left unchanged. If, on the other hand, the short-term relative motion measure is less than the low threshold, then the case is switched to the new one. Going back to the long-term relative measure check, if it is smaller than the R_l threshold, then the case might not be performing that well and, therefore, the short-term relative measure is compared against a more strict threshold – R_s threshold. If it falls below the threshold, then the new case is selected. Otherwise, the case spatial similarity is compared against a strict threshold – S_{high} threshold. If the similarity is less, then the new case is selected, otherwise the current case is given more time to exert itself.

4) Case Adaptation

If it is decided at the previous step to keep the current case, then this step is not executed. If it is decided, however, to apply a new case, then the new case needs to be fine-tuned to the current environment.

The adaptation algorithm is very simple:

X = ($R_{l\ adapt}$threshold + $R_{s\ adapt}$threshold) / (R_l + R_s);
Y = $R_{l\ adapt}$threshold / R_l ;
Z = $R_{s\ adapt}$threshold) / R_s;
If (R_l < $R_{l\ adapt}$threshold and R_s < $R_{s\ adapt}$threshold)
    Increase *Noise_Gain* proportionally to X;
    Increase *CaseTime* proportionally to X;
    Limit *Noise_Gain* and *CaseTime* from above;
Else if (R_l < $R_{l\ adapt}$threshold)
    Increase *Noise_Gain* proportionally to Y;
    Increase *CaseTime* proportionally to X;
    Limit *Noise_Gain* and *CaseTime* from above;
Else if (R_s < $R_{s\ adapt}$threshold)
    Increase *Noise_Gain* proportionally to Z;
    Limit *Noise_Gain* from above;
End;

The adaptation algorithm looks at both the long-term and short-term motion measures of the robot and increases the level of noise in the robot's behavior if any of the measures fall below the corresponding thresholds. The amount to which the noise is increased is proportional to how long the robot's progress was impeded as determined by the long- and short-term motion measures. If a robot was lacking progress for long enough, then the long-term motion measure R_l falls below its threshold and the *CaseTime* threshold is also increased to assure that the new case is applied long enough for the current environment.

After the case is adapted it is applied. The application is simply the extraction of the behavioral assemblage parameters from the case and passing them to the behavioral control unit within the *MissionLab* system.

III. Simulation Results

The system was first tested in a simulated environment. *MissionLab* provides a simulator as well as logging capabilities allowing the collection of the required statistical data easily.

Figure 6 shows the runs of a simulated robot with and without a CBR unit.

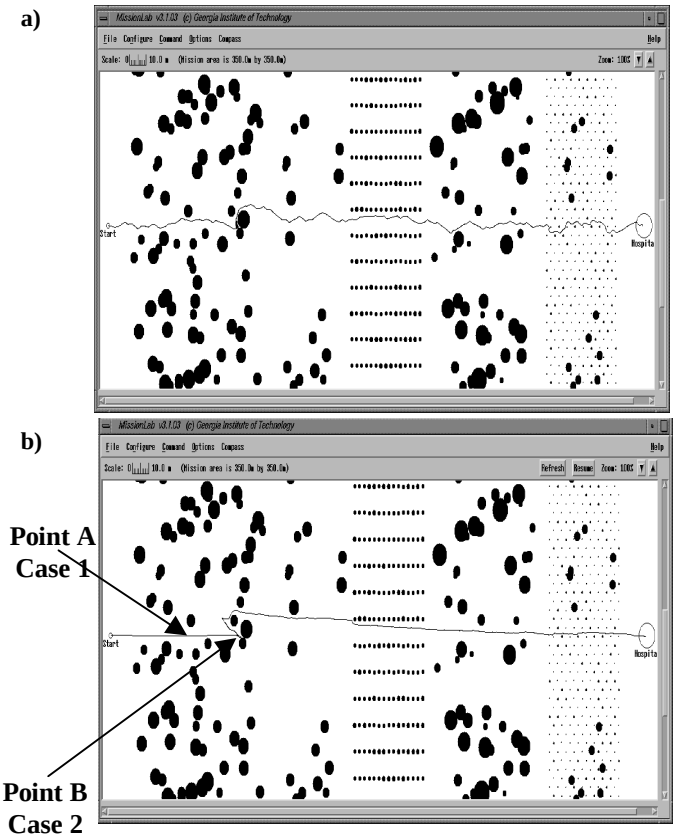

Figure 6. Robot runs in simulated environment. a) without CBR module; b) with CBR module;

During the entire run the same behavioral assemblage is used. However, as the environment changes from one type to another, the CBR module re-selects the set of parameters that control the behavioral assemblage. As a result, if CBR is disabled a robot requires a higher level of noise in its behavior in order to complete the mission (figure 6a). If, however, the CBR module is enabled, then **Wander** behavior is rarely used, and the distance traveled by the robot in Figure 6b is 23.7% shorter, whereas mission completion time is 23.4% less than in Figure 6a. For example, during the part of the run before the local minimum produced by two obstacles is encountered (point B in figure 6b) the robot uses a case 1 called CLEARGOAL case (figure 7b left). In this case no noise is present in the robot behavior making the trajectory a straight line. When the robot approaches the two obstacles, it switches to the case 2 called FRONTOBSTRUCTED_SHORTTERM

a)

Environment characteristics at A:	Environment characteristics at B:
Spatial Vector:	**Spatial Vector:**
D (goal distance) = 300	*D* (goal distance) = 275
density distance	density distance
Region 0: $\sigma_0 = 0.31$; $r_0 = 5.13$	Region 0: $\sigma_0 = 1.00$; $r_0 = 0.11$
Region 1: $\sigma_1 = 0.71$; $r_1 = 2.83$	Region 1: $\sigma_1 = 0.79$; $r_1 = 0.11$
Region 2: $\sigma_2 = 0.36$; $r_2 = 7.03$	Region 2: $\sigma_2 = 0.38$; $r_2 = 0.12$
Region 3: $\sigma_3 = 0.54$; $r_3 = 2.80$	Region 3: $\sigma_3 = 1.00$; $r_3 = 0.11$
Temporal Vector:	**Temporal Vector:**
(0 - min, 1 - max)	(0 - min, 1 - max)
ShortTerm_Motion $R_s = 1.000$	ShortTerm_Motion $R_s = 0.010$
LongTerm_Motion $R_l = 0.931$	LongTerm_Motion $R_l = 1.000$
Traversability Vector:	**Traversability Vector:**
(0-untraversable, 1- excellent)	(0-untraversable, 1- excellent)
f_0=0.92 f_1 =0.58 f_2 =1.00 f_3=0.68	f_0=0.02 f_1 =0.22 f_2 =0.63 f_3=0.02

b)

Case 1 used at A:	Case 2 used at B:
CLEARGOAL	**FRONTOBSTRUCTED_SHORTTERM**
Spatial Vector:	**Spatial Vector:**
D (goal distance) = 5	*D* (goal distance) = 5
density distance	density distance
Region 0: $\sigma_0 = 0.00$; $r_0 = 0.00$	Region 0: $\sigma_0 = 1.00$; $r_0 = 1.00$
Region 1: $\sigma_1 = 0.00$; $r_1 = 0.00$	Region 1: $\sigma_1 = 0.80$; $r_1 = 1.00$
Region 2: $\sigma_2 = 0.00$; $r_2 = 0.00$	Region 2: $\sigma_2 = 0.00$; $r_2 = 1.00$
Region 3: $\sigma_3 = 0.00$; $r_3 = 0.00$	Region 3: $\sigma_3 = 0.80$; $r_3 = 1.00$
Temporal Vector:	**Temporal Vector:**
(0 - min, 1 - max)	(0 - min, 1 - max)
ShortTerm_Motion $R_s = 1.000$	ShortTerm_Motion $R_s = 0.000$
LongTerm_Motion $R_l = 0.700$	LongTerm_Motion $R_l = 0.600$
Traversability Vector:	**Traversability Vector:**
(0-untraversable, 1- excellent)	(0-untraversable, 1- excellent)
f_0=1.00 f_1 =1.00 f_2 =1.00 f_3=1.00	f_0=0.14 f_1 =0.32 f_2 =1.00 f_3=0.32
Case Output Parameters:	**Case Output Parameters:**
MoveToGoal_Gain = 2.00	*MoveToGoal_Gain* = 0.10
Noise_Gain = 0.00	*Noise_Gain* = 0.02
Noise_Persistence = 10	*Noise_Persistence* = 10
Obstacle_Gain = 2.00	*Obstacle_Gain* = 0.80
Obstacle_Sphere = 0.50	*Obstacle_Sphere* = 1.50
Bias_Vector_X = 0.00	*Bias_Vector_X* = -1.00
Bias_Vector_Y = 0.00	*Bias_Vector_Y* = 0.70
Bias_Vector_Gain = 0.00	*Bias_Vector_Gain* = 0.70
CaseTime = 3.0	*CaseTime* = 2.0

Figure 7. a) Environment features at points A (left) and B (right); b) Cases used at point A (left) and point B (right).

(figure 7b right). In this case, the gains of the **Wander** and **BiasMove** schemas and *Obstacle_Sphere* are increased. This ensures that the robot quickly gets out of the local minima and proceeds toward the goal switching back to the CLEARGOAL case.

Figure 8 graphs the statistical data gathered in the simulations. The performance represented by traveled distance and time steps was measured as a function of obstacle density. Just as in the figure 6, the robot had to travel through different types of environments, but the average density varied across trials. Note that for the runs without the CBR module, the optimal set of parameters was chosen for a given average obstacle density. This was equivalent to a user specifying the optimal parameters for a given mission. Even larger improvement could be expected if the parameters were chosen constant throughout all the trials. As seen in figure 8, if the average obstacle density is very small (below 12%), then the improvement is insignificant. This is due to the fact that in an environment that is almost obstacle-free, there really is only one case applied all the time. The same set of parameters can be chosen manually for the robot without the CBR module. As the obstacle density increases, however, the cases are switched more and more often leading to a significant improvement in performance.

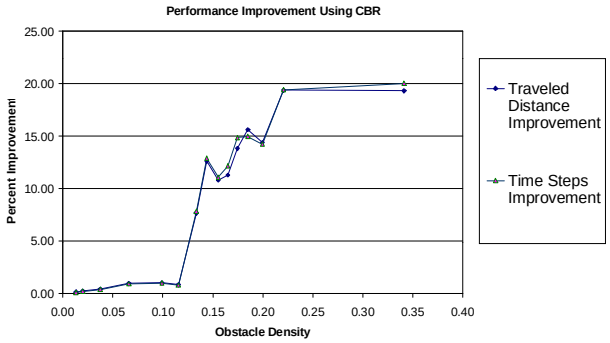

Figure 8. Statistical evaluation of the performance improvement of the system with CBR over the system without CBR (in simulations).

IV. Robot Experiments

The system was also tested on a real robot, a Nomad 150 series robot. It had 12 sonar sensors evenly placed around it. The information from these sensors was the only perceptual input driving the behavior of the robot. The *MissionLab* system described earlier provides support for real robotic systems including the Nomad 150 robots. Thus, for the real robot experiments the exact same framework as for the simulations was used.

The environment for the real robot experiments is shown in Figure 10. The chairs were used to introduce additional obstacles in the environment. The tree in the white vase by the couch shown in the back of the picture represents the goal for the robot.

Figure 10a shows the start of the robot run. The path is clear and a traversability vector would indicate that the 0^{th} region directed toward the goal has full traversability. This corresponds to the CLEARGOAL case (figure 7b left) with zero **Wander** schema gain, and the robot moves straight toward the goal. As it reaches the small box canyon constructed by the three chairs (figure 10b), a traversability vector indicates little traversability of the 0^{th} region and high traversability for other regions. The new case - FRONTOBSTRUCTED_SHORTTERM (figure 7b right) - is applied with greater gain for the **Wander** schema, larger sphere of influence for obstacles and some gain for the **BiasMove** schema directing the robot back from the chairs. As a result, the robot comes quickly out of the canyon. In figure 10c the robot is again clear to the goal, and a case with no **Wander** behavior is selected that makes the robot go straight to the goal.

Ten runs were conducted with the CBR module and ten without. Each pair of runs was done on exactly the same environment. Just as in simulations, the trials ranged

Improvement(%)		
Obstacle Density	Traveled Distance	Time Steps
Low	6.2	3.3
Medium	17.8	17.6
High	26.4	28.6

Figure 9. Improvement in robot performance with CBR module versus without the CBR module (in real robot experiments)

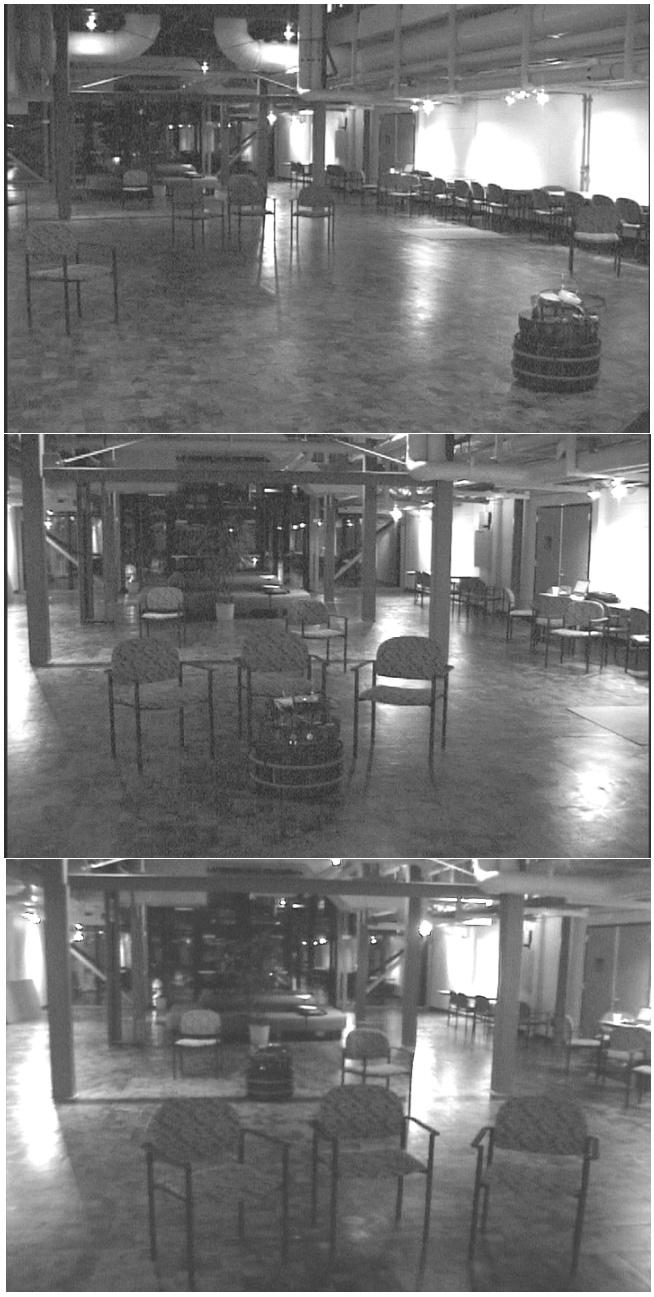

Figure 10. Real robot run. Chairs are used as obstacles; the tree in the back by the couch is the goal of the robot.

from very low obstacle density environment to a quite large obstacle density. The collected data is shown in the figure 9. The numbers correlate well with the simulation-based data also showing that as the average obstacle density increases, the benefits from the CBR module also increase.

V. SUMMARY

This paper presented a robotic system that incorporated case-based reasoning into the process of selection and modification of parameters that control the behavioral assemblage in a schema-based navigational system. The CBR module allows for an easier design of behavioral assemblages. First, the behavioral parameters do not need to be chosen carefully any longer since the CBR module sets them up automatically in real-time based on the current robot's environment. Secondly, fewer behavioral assemblages need to be designed by virtue of the fact that by adjusting the parameters the CBR module in effect selects different behavioral assemblages that have the same set of active schemas. The CBR module was designed in such a way as to provide very robust performance in case selection and adjustment processes. The simulation and real robot experiments clearly showed significant improvement in the robot navigational tasks.

Future work includes the addition of automatic learning of new cases through experience and the extension of the CBR module application beyond the navigational tasks. Also, the integration work of the CBR module within a larger framework of learning algorithms within *MissionLab* is planned for the future.

VI. ACKNOWLEDGEMENT

This research is supported by the DARPA MARS program under contract #DASG60-99-C-0081. The authors of the paper would like to thank the following people who were invaluable in the work with the *MissionLab* and real robot experiments: Douglas MacKenzie, Tucker Balch, Yoichiro Endo, William Conrad Halliburton, Mike Cramer, and Dr. Tom Collins.

REFERENCES

[1] A. Ram, R. C. Arkin, K. Moorman and R. J. Clark, "Case-based Reactive Navigation: a Method for On-line Selection and Adaptation of Reactive Robotic Control Parameters," *IEEE Transactions on Systems, Man and Cybernetics - B*, 27(30), pp. 376-394, 1997.

[2] A. Ram, J. C. Santamaria, R. S. Michalski and G. Tecuci, "A Multistrategy Case-based and Reinforcement Learning Approach to Self-improving Reactive Control Systems for Autonomous Robotic Navigation," *Proceedings of the Second International Workshop on Multistrategy Learning*, pp. 259-275, 1993.

[3] C. Vasudevan and K. Ganesan, "Case-based Path Planning for Autonomous Underwater Vehicles," *Autonomous Robots*, 3(2-3), pp. 79-89, 1996.

[4] M. Kruusmaa and B. Svensson, "A Low-risk Approach to Mobile Robot Path Planning," *Proceedings of the 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, 2, pp. 132-141, 1998.

[5] P. Gugenberger, J. Wendler, K. Schroter, H. D. Burkhard, M. Asada and H. Kitano, "AT Humboldt in RoboCup-98 (team description)," *Proceedings of the RoboCup-98*, pp. 358-363, 1999.

[6] M. M. Veloso and J. G. Carbonell, "Derivational Analogy in PRODIGY: Automating Case Acquisition, Storage, and Utilization," *Machine Learning*, 10(3), pp. 249-278, 1993.

[7] S. Pandya and S. Hutchinson, "A Case-based Approach to Robot Motion Planning," *1992 IEEE International Conference on Systems, Man and Cybernetics*, 1, pp. 492-497, 1992.

[8] P. Langley, K. Pfleger, A. Prieditis and S. Russel, "Case-based Acquisition of Place Knowledge," *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 344-352, 1995.

[9] D. Mackenzie, R. Arkin, and J. Cameron, "Multiagent Mission Specification and Execution," *Autonomous Robots*, 4(1), pp. 29-57, 1997.

[10] R. Arkin and T. Balch, "AuRA: Principles and Practice in Review," *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2), pp. 175-189, 1997.

[11] R. Arkin, "Motor-Schema based Mobile Robot Navigation," *International Journal of Robotics Research*, 8(4), pp. 92-112, 1989.

[12] J. Kolodner, *Case-Based Reasoning*, Morgan Kaufmann Publishers, San Mateo, 1993.

[13] N. Chalmique Chagas and J. Hallam, "A Learning Mobile Robot: Theory, Simulation and Practice," *Proceedings of the Sixth Learning European Workshop*, pp.142-154, 1998.