Final Report : DARPA MARS 2020 Program
NASA JSC GRANT # NAG9-1446

Acquisition of Autonomous Behaviors by Robotic Assistants

Kazuhiko Kawamura, PI

Co-PIs:
R.A. Peters II, N. Sarkar, R.E. Bodenheimer

Ph.D. Graduate Students:
E. Brown, C. Campbell, K. Hambuchen, C. Johnson, A. B. Koku P. Nilas, J. Peng, P. Ratanaswasd, T. Rogers, and J. Rojas

Masters Graduate Students:
K. Achim, C. Clifton, W. Dodd, D. Erol, S. Ferguson, P. Fleming, S. Gordon, H. Kaymaz-Keskinpala; T. Keskinpala, L. Ma, R. Olivares, A. Spratley, and C. Zhou

***Undergraduate Students*:**
A. Choe, A. Cook, C. Costello, D. Martinez, S. Renkes, and R. Sunderland

Center for Intelligent Systems
Vanderbilt University
Nashville TN 37235-0131
http://eecs.vanderbilt.edu/CIS

December 2004

This report summarizes the work conducted at the Vanderbilt Cognitive Robotics Lab and NASA Johnson Space Center in the scope of the DARPA MARS 2020 Program Project "Acquisition of Autonomous Behaviors by Robotic Assistants", via the NASA subcontract NAG9-1446. It covers the full period of the project from August 1, 2002 through the no-cost extension ending on December 31, 2004.

TABLE OF CONTENTS

1. Introduction

The Vanderbilt team performed research towards achieving autonomous robotics assistants, with a focus on cognitive applications using Vanderbilt's ISAC [Kawamura 2004] and NASA's Robonaut humanoid robots [Ambrose 2000]. Our approach consists of coupling perception-level behavior learning with high-level cognitive control and learning. We have achieved the following results:

- The *Sensory EgoSphere (SES) software* transferred and integrated on Robonaut
- A proof-of-concept experiment in *superpositioning of behaviors learned through teleoperation* conducted on Robonaut
- An *intelligent health monitoring system* developed and tested on Robonaut
- A proof of concept experiment in *cognitive control using a working memory and modular control* developed and tested on ISAC
- *SES- and SMC-based mobile robot navigation* conducted on Segway.

These topics are presented in the following sections of this report.

2. The Sensory EgoSphere

2.1 Background

Today's robots can be equipped with a wide and powerful array of sensing modalities, but their cognitive abilities are still primitive. Coordinating input from the sensors presents significant challenges for robot cognition, and can even be confusing to a human supervisor or teleoperator. Under the original MARS grant, a mediating interface, called the *Sensory EgoSphere* (SES), that serves to coordinate sensory information for cognitive processing was developed [Kawamura 2000] [Hambuchen 2004]. The SES serves as an attentional, associative, short-term memory in the robot's control system. It operates asynchronously as a high-level agent in a parallel, distributed, object-oriented or agent-based, control system that includes independent, parallel sensory processing modules.

For a person interacting with the robot, either through teleoperation or as a supervisor, the SES can be visualized as a spherical shell centered on the robot's base frame. Each point on the shell is a locally connected memory unit with an associated activation vector and a temporal decay. From an internal, computational point of view, the SES is a graph whose edges form a geodesic tessellation of a sphere. Each node of the graph connects to a database in addition to its neighbors. An SES manager program interacts with other agents to write and read information to the SES.

The SES useful for people interacting with a robot as it provides an egocentric representation of the robot's knowledge of the current environment. Additionally though, the SES possesses features that allow the cognitive mechanisms of a robot to leverage and coordinate sensor data easily.

2.2 SES Overview

There are two primary hypotheses behind our definition and use of the SES: (1) often, a physical event in the environment will stimulate more than one of the robot's sensors, and (2) changes in motion of the robot will precipitate sensor events.[2] Thus, if two or more of the SPMs detect events at nearly the same time, and if directionally sensitive modules report their events as having emanated from similar directions in space, then we presume that the robot has detected a real event. Moreover, if a change in motion is accompanied by the registration of events by more than one sensor we presume the events may be relevant. By including proprioceptive sensing and motor control sequences with the exteroceptive sensory streams that project to it, the SES makes spatio-temporal sensory-motor data associations. It does so without having to perform any comparative operations on the sensory signals.

The SES can be defined mathematically as the set of radial distances from a designated point on the robot to the first encountered object points in space. That definition is simple but incomplete. This purely geometric definition implies that the structure is memory-less, whereas in fact it can be used as a memory structure. Much more than the distance to the first object in space is stored on the SES.

In its implementation on a robot, the SES is a database with associated computational routines. It is a sparse map of the world that contains pointers to sensory data or descriptors of objects or events that have been detected recently by the robot. Given that the sensors on a robot are discrete, there is nothing to gain by defining the SES to be a continuous structure. Moreover, the computational complexity of using the SES increases with its size which is, in turn, dependent on its density (number of points on its surface). The database's graphic connectivity is isomorphic to a regular, triangular tessellation of a sphere centered on the coordinate frame of the robot.

2.2.1 Geodesic Dome Topology

We define the topological structure of data connectivity in the SES to be that of a geodesic dome, shown in Figure 1, since it is a quasi-uniform triangular tessellation of a sphere into a polyhedron [Edmondson 1987] [Urner 1991] [Albus 1991]. The triangles connect at vertices forming 12 pentagons and a variable number of hexagons. The pentagons are evenly distributed so that the node at the center of one is connected to the centers of five others by ~ vertices, where ~ is called the frequency of the dome.

Two questions naturally arise. First, why not use a fully 3-D representation like an occupancy grid. Second, why use a fixed tessellation?

From the robot's point of view, it is stationary while the world moves. Features of the environment are located in various directions, at various depths. Only directional information is needed to place features within the robot's locale. Knowledge of depth is needed only for specific interaction with an object.

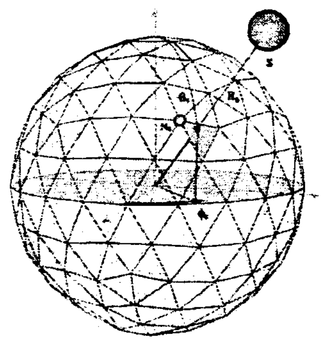


Figure 1. A geodesic dome representation of the SES.

This representation maintains direction but computes depth as only needed. We believe this representation is more efficient than filling an occupancy grid with data whose distance may not be needed. The fixed size of the tessellation provides a fixed number of entry points into the sensory-motor database. The SES manager maintains a direct mapping between directions with respect to the robot and SES nodes. Only that mapping must be updated with motion of the robot. The data itself does not require modification until it is accessed, at which time an estimate of its current location can be updated. The search for a specific object or sensory feature in the worst case requires the traversal of only a fixed number of nodes.

2.2.2 Data Structure

The geodesic dome topology of the SES organizes sensory and motor data with respect to a locally connected graph of pointers to data structures, indexed by location. Data from a directional SPM is stored at the node that is closest to the direction from which the stimulus arrived. One pointer exists for each vertex on the dome. Each pointer has six or seven links, one to each of its five or six nearest neighbors and one to a tagged-format data structure. The latter comprises a terminated list of alphanumeric tags each followed by a spatial location, a time stamp, an activation level, and another pointer. The fixed size, local connectivity, and directional indexing simplifies both the storage of sensory-motor data and ego-centric searches for it (see Figure 2).

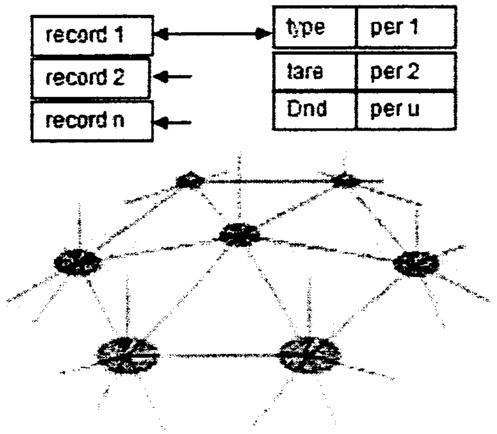


Figure 2. Each node points to its neighbors and to a database record that references other records of various types.

The tags describe the data modality, a description of the data, the data's full-resolution spatial location and the name of the feature or object that the data represents. The spatial location is the estimated direction of the data source or object and, if available, the distance to it. A time stamp designates when the data was registered onto the SES. The activation indicates the relative importance of the specific data. The pointer associated with the tag holds the location of a structure that contains (or points to) the sensory data. The number of tags and their types on any vertex of the dome are completely variable. A central node, connected to all of the others from the center of the sphere, monitors the time sequence of sensory and motor inputs to enable temporal association of spatially distributed events or non-directional events. The node also contains tags to the data, associated time stamps and activations of non-directional sensory data.

Sensory processing modules write information to the SES through a software object, the SES manager, which in turn interfaces to a standard database such as MySQL [Yarger 1999]. The manager can also perform a breadth-first search of the SES for the vertices that contain a given tag. The software object requesting the search can specify various search parameters such as the starting location, number of vertices to return, search depth, etc. The fixed number of nodes keeps the search paths fixed as the amount of data on the sphere increases.

2.3 Motion Transformations of the SES

If a robot and its environment are stationary, then the locations of data will not move on the SES. If the base frame of the robot remains fixed in space over time, any articulated motion of the robot can be counteracted via its known kinematics (see Figure 3). To correctly register moving objects on a stationary SES requires object tracking, and thus prediction and searching. Moreover if the base frame of the robot moves, the locations of data on the SES will also move both as functions of the heading and velocity of the robot and of the distances of the sensed objects from the robot. Thus, as the robot moves, the node locations of data on the SES must be shifted.

Purely rotational motion of the base frame is easily compensated for by oppositely rotating the SES. That aligns the SES with the environment while the robot moves within the SES. Translational motion of the base frame requires that object locations on the SES be shifted as a function of their distance from the base frame. Such shifting of the information is prone to error. This error is not critical since the estimated SES location of an object serves as the starting point for a sensory search of the environment to locate the object more exactly. In this capacity, the SES provides reasonable starting locations for searches, reducing the time necessary to track multiple objects.

As a robot moves, objects in its environment shift relatively. Thus, projections of objects move on the SES. If the robot's motion is known, the change in SES projection can be estimated for any object a known distance away. If the distance to an object is unknown, motion on the SES permits the distance to be estimated. Reciprocally, the concerted motion of a set of objects on the SES enable the robot to compute its motion relative to the objects. These estimation procedures also can alert the robot to independent motion by an object.

There are three types of motion transformation typically employed in the SES: pure translation, as when a mobile robot is traveling along a straight path; translation coupled with rotation, the most general case of ego-sphere motion; and articulated motion, as when a humanoid robot exercises it end-effectors with respect to its base frame. For implementation purposes, the details and description of these motions are contained in [Peters 2003] [Hambuchen 2004].

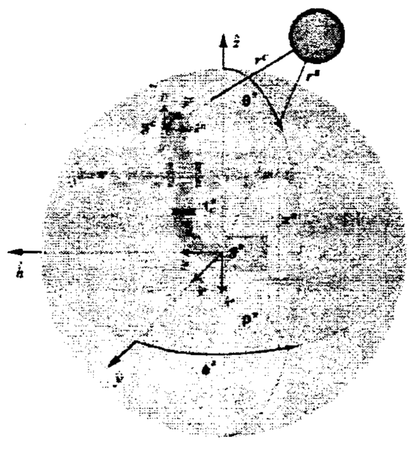


Figure 3. Robonaut, an articulated humanoid, within its SES.

2.4 Applications and Results

In this section, we present both quantitative and qualitative results when the SES is used as a short-term memory, for spatial localization and navigation, for coincidence detection, and as an attentional mechanism. These results were obtained from implementations of the SES on Vanderbilt's humanoid robot, ISAC [Kawamura 2000] on NASA's humanoid robot, Robonaut [Ambrose 2000] and on a mobile robot at Vanderbilt [Kawamura 2002]. These robots have sensory processing and motor control modules that operate in parallel continuously, independently, and asynchronously; additionally they communicate through message passing [Peters 1999]. The SES is most effective when implemented on robotic architectures possessing these capabilities.

2.4.1 Short-Term Memory

As a short-term memory (STM), the SES is useful for maintaining an inventory of objects in the robot's locale for subsequent manipulation or other action. The SES is currently being used in that way by ISAC at Vanderbilt, Robonaut at NASA-JSC, and Cog at MIT. When the robot recognizes an object, the location of a point of reference on the object (part of the object definition) and the object's pose are stored along with an identifier and time stamp at the closest SES node. The identifier is used as a tag by the SES for its search and recall routines. The time stamp can be used along with an activation decay constant to compute a probability that the object is at the recorded location after time has elapsed.

As the robot, its environment, or the object moves, its location is updated by the SES so that the robot always stores the object's position relative to the base frame. This position is likely to accrue errors if the robot is not actively tracking the object with its sensors. The SES, however, provides the starting location for a sensory search if the object is not found by the sensors at the recorded location upon later recall. The SES also maintains locations of objects if motion results in the occlusion of one object by another. The spatial layout of the SES keeps track of the spatial relationships between objects so that the robot can know "what is where" [Peters 2003].

Because the SES manager rotates the nodes and shifts the data to compensate for motion of the robot's base frame, data from specific locations in the environment accumulate over time. Object recognition agents designed to monitor the SES can periodically analyze the data accumulating at a location. If the data is consistent with a known object the agent can tag the location with an object label and a confidence level.

2.4.2. Sensory-Motor Data Association

If parallel SPMs output to the SES, it can associate sensory and motor data through spatio-temporal coincidence detection. Figure 2 shows an example of a node and its immediate neighbor nodes. Note: the data structure of the SES used by Cog was developed independently and differs from that described here.

The activation values of the nodes indicate their relative importance at the current time. Each node contains a radial basis function (RBF) that spreads activation to its neighbors according to equations (2)-(3) above. If multiple data are registered in the same area at about the same time, activation will increase around a central node. For 1-D sensors, registration occurs only at the equatorial axis of the SES. Therefore, activation must be spread longitudinally so that events co-occurring away from the equatorial axis may overlap with the 1-D sensor events. Upon registration of data from a 1-D sensor, nodes along the longitude closest to the registration angle each receive activation as if they were the original node. Figure 4 shows the sensory processing overview.

To perform coincidence detection, the node with the highest activation is selected. All data that contributed to the activation of that node is retrieved from the SES. Temporal coincidence can be detected using processing latencies of the SPMs, which can be measured experimentally. The latencies define a time interval during which all sensory events are considered to be simultaneous.

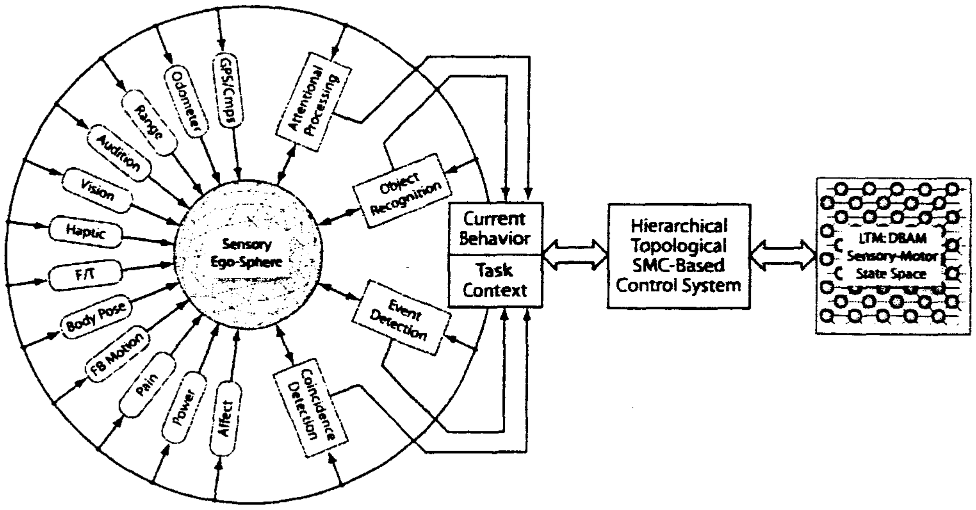


Figure 4. Sensory processing overview [Peters 2003].

Tests of coincidence detection for sensory data association involved recognition of objects that were uniform in color, that were movable, and that could make sounds (i.e., an orange rattle, and a purple toy that talks). Objects were individually presented to the Vanderbilt humanoid, ISAC, producing 32 separate sets of multi-modal events. ISAC's stereovision head can detect the angular position of an object. Sonic localization and IR motion detection are far less accurate. Thus, the angular position of an event that produces sound, motion, and imagery may be grossly mismeasured by those two sensors. In 12 of the 32 trials, the error in measurement of both sound and IR was not detectable as a coincidence [Hambuchen 2004].

An experiment comprising 21 trials was performed with two sources presented to the robot in succession at multiple locations. Each source generated three separate events (visual, IR, sound). In all of the trials, all co-occurring events from one source were selected. The cumulative time latency for visual, IR and sound events averaged three seconds while the time range used in coincidence detection was four seconds.

Eleven objects were individually presented to the Vanderbilt humanoid, ISAC. Each object produced visual data from color segmentation, motion data from IR motion sensors, and sound data from sound localization. The resolution of sensors were vision, IR motion, and sound. The neighborhood can be increased or decreased depending on the resolution of the SPMs that send data to the SES.

An experiment comprising 40 trials was performed with multiple sources presented to the robot. The source that produced the most data (visual, sound, motion) was always selected. When all SPMs reported correctly, all co-occurring data were always selected. As the object separation approached the resolution of the sensors, incorrect results were reported.

2.4.3 Attention

The nodal activation vectors of the SES can be used to direct the attention of the modules that read data from the SES and, thereby, the attention of the robot. The SES can be biased toward the selection of specific data by modulating the strength of activations assigned to SES nodes. This bias is useful for directing the robot's attention during tasks such as picking up tools, or during contextual circumstances such as working with people. The attention network balances the trade-off between contextually important data and unexpected yet salient data.

The attention network combines the activation from the nodal RBFs to represent salience data in the environment with priority values that represent desired data. The focus of attention (FOA) is selected as the node that receives the highest combination of activation from both the RBFs and the priority values. This node is then sent through coincidence detection to determine which data originated from the same source.

Initial experiments have been performed on ISAC to determine at what level priority values shift the focus of attention from desired data to salient data. In these trials, the desired task for the robot was to grasp a turquoise beanbag. At 15-20 seconds after visual detection of the beanbag, a person moved back and forth in another area of the environment while clapping her hands, producing both sound and motion data. Priority values were decreased from 5.0 to 0.1. During all trials, the beanbag was selected as the focus of attention. Since the vision sensors have a small resolution and high reliability, one visual event creates the same amount of activation from the RBF as a sound event and a motion event combined. Therefore, trials were repeated to include visual data from the person. Priority values were again decreased from 5.0 to 0.1. The beanbag was selected as the FOA until the priority value reached 1.0. At this point, the motion, sound, and person were selected as the FOA.

3. Behaviors Learned through Teleoperation

3.1. Approach

The approach described here builds on the self-organization of sensory-motor information in response to a robot's actions within a loosely structured environment. In [Pfeifer 1997], Pfeifer reported that sensory data and concurrent motor control information recorded as a vector time-series formed clusters in a sensory motor state-space. He noted that the state-space locus of a cluster corresponded to a class of motor action taken under specific sensory conditions. In effect, the clusters described a categorization of the environment with respect to sensory motor coordination (SMC).

An exemplar of an SMC cluster corresponds at once to a basic-behavior (as used by Brooks [Maes 1990] and later defined by [Matarić 1992]) and to a competency module in a spreading activation network. Thus, if a robot is controlled through an environment to complete a task while recording its SMC vector time-series, the result is a state-space trajectory that is smooth during the execution of a behavior but that exhibits a corner or a jump during a change in behavior (an *SMC event).* From this, a DEDS description of the task can be formed as a sequence of basic behaviors and the transitions between them. The task is learned in terms of the robot's own sensors, actuators, and morphology.

This section reports the results of learning to reach toward and grasp a vertically oriented object at an arbitrary location within the robot's workspace by superpositioning a set of SMC state space trajectories that were learned through teleoperation. The ideas behind the procedure are based on a number of assumptions:

1. When a teleoperator performs a task it is her/his SMC that is controlling the robot. So controlled, the robot's sensors detect its own internal states and those of the environment as it moves within it. Thus the robot can make its own associations between coincident motor actions and sensory features, as it is teleoperated.

2. In repeating a task several times, a teleoperator will perform similar sequences of motor actions whose dynamics will depend on his/her perception of similar sensory events that occur in similar sequence. As a result, the robot will detect a similar set of SMC events during each trial. Therefore each trial can be partitioned into *SMC episodes,* demarcated by the common SMC events.

3. Sensory events that are salient to the task will occur in every trial; sensory signals that differ across trials are not significant for the task and can be ignored. By averaging the time-series for each episode point-wise over the trials, a canonical representation of the motor control sequence can be constructed. As a result of the averaging, true events in the sensory signals will be enhanced and those that are random will be suppressed.

This approach does not form an approximation of the inverse kinematics of the manipulator. Rather, it learns 6-axis spatial end-effector trajectories that are sent as position commands to the robot, which computes its own inverse kinematics. This somewhat higher level approach extracts Cartesian motion and pose trajectories, finger position trajectories, and sensory state information to create a sensory-motor (or, perhaps more accurately, a sensory-motion) description of the task.

3.2 Behavior Superposition

3.2.1 Data Collection

There were four phases in the data gathering and analysis for this learning task (Figure 5):

1. A teleoperator controlled the robot through the tasks that would serve as examples. Five trials at each of nine locations were performed of a reach and grasp of a vertically oriented object (a wrench). As the teleoperator performed these example motions, Robonaut's sensory data and motor command streams were sampled and recorded as a vector time-series or signal.
2. The SMC events common to all trials were found and used to partition the signal into episodes. The episodes were time-warped so that the *j*th episode in the *k*th trial had the same duration (and number of samples) as the *j*th episode in every other trial.
3. The signals were averaged over all five trials at each location to produce a canonical, sensory-motor data, vector time-series for each location. This approach is similar both to that of Jenkins and Matarić [Jenkins 2002] and to those analyzed by Cohen [Cohen 2001].
4. These generalized motions were combined using the process described by Rose et al. [Rose 1998] called *Verbs and Adverbs*.

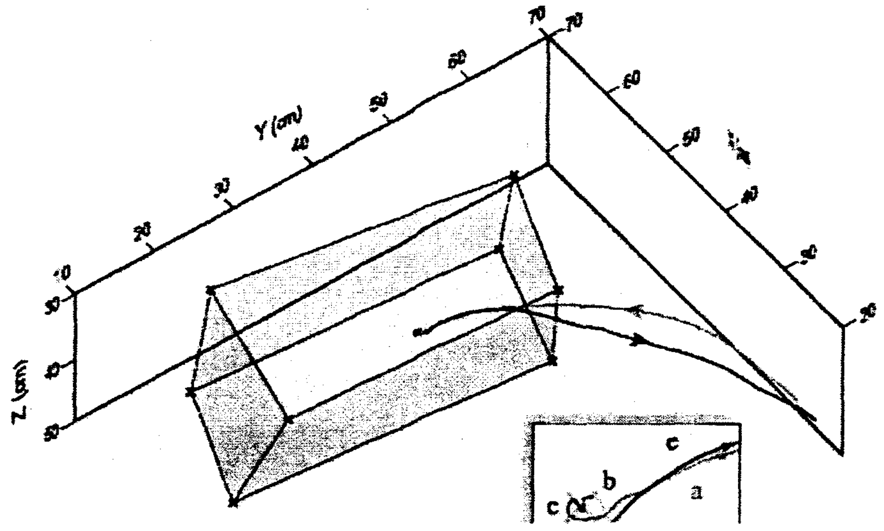


Figure 5. Plot of the 9 exemplar object locations from the robot's viewpoint. The contour is the end-effector trajectory from one trial of the experiment where the object was at the ninth position, in the center of the box. The inset shows the grasp, hold, and release episodes in greater detail. They are: (a) reach, (b) grasp, (c) hold, (d) release, (e) withdraw.

3.2.2. Segmentation
The time-series data from the experiment was manually segmented into 45 trials according to markers embedded in the voice channel of the robot's data stream. Then each trial was partitioned into five SMC episodes[3] (reach, grasp, hold, release, withdraw) demarcated by SMC events that were found through an analysis of the mean-squared velocity (MSV) of the joint angles. The factor of 15 was used for the upper threshold because it yielded the number of episodes that were expected (Figure 6).

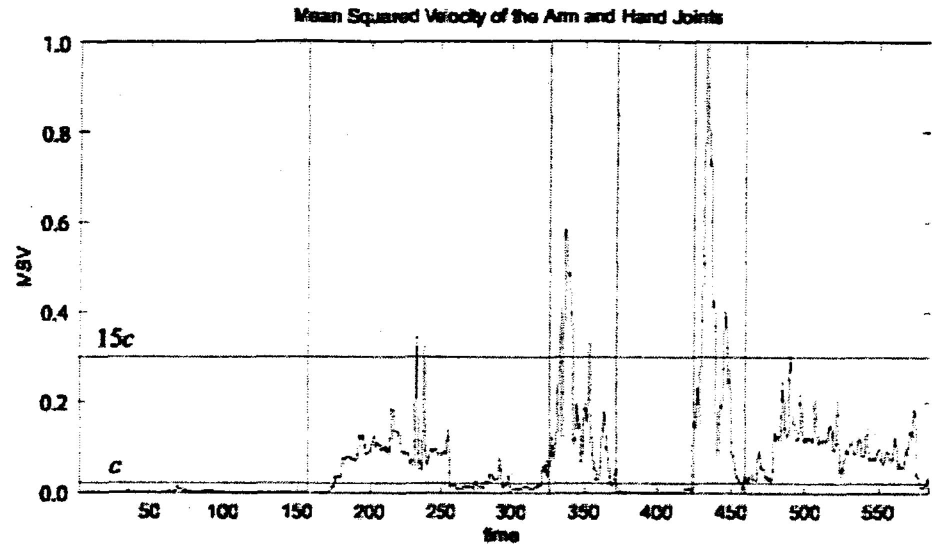


Figure 6. Plot of an instantaneous mean-squared joint velocities, z, for one teleoperated trial at one object location. The lower and upper thresholds, c and $15c$, are indicated by horizontal lines. Episode boundaries (pre-motion, reach, grasp, hold, release, withdraw) are demarcated by vertical lines.

The MSV was found to be an excellent indicator of the grasp, hold, and release events if the hand joint velocities were included in it. It was not reliable in detecting those events if only the arm joint velocities were included. The vector time series between two SMC events were taken as SMC episodes that corresponded to distinct behaviors.

3.2.3. Time Warping: Normalization and Averaging
Once the segmentation of the data was complete, the SMC episodes that comprise the task were time-warped through resampling to have a duration equal to the average duration of the 45 trial episodes. Then for each of the 9 locations the average vector time-series was computed from the five corresponding trials. For example, the *reach* behavior averaged 150 time steps across the 45 trials. Each of the time-series that comprised the *reach* episodes was time-warped and resampled to have length 150. The five *reach* episodes from the five trials at each location were averaged to create nine exemplar *reach* episodes each with 150 samples in duration. Fig. 5 shows the trajectories from the five trials at one location and the average of the five.

In these experiments we used a point-wise linear averaging of the time-normalized sensory-motor episodes to produce an exemplar for the task. The effect of averaging the five trials at each location was to enhance those characteristics of the sensory and motor signals that were similar in the five and to diminish those that were not. One could use the median value at each point, if a minority of the exemplars showed deviations due to noise or other mismeasurement. Moreover, signals that exhibit nonlinear behavior with respect to time (e.g. a binary or on/off signal) would require a median or other order-statistic filter to preserve the signal characteristics. Certainly, averaging would produce a skewed result if one of the exemplar episodes were significantly different from the others to be combined with it. However, it was a premise of this work that such episodes would not differ significantly from each other in their salient features. If that premise were incorrect, the characterization of a behavior through the type of analysis described here would be of dubious value. But, the premise was found to hold throughout these particular experiments.

Through the four-step procedure nine sensory-motor state-space trajectories were created. These were taken to be the exemplars of the clusters formed by the five trials of the reach-and-grasp task at each of the nine locations. More specifically, in our teleoperational experiment, we completed the following steps:

1. Teleoperate Robot through manipulation task
2. Extract canonical SMC or control policy description of task as a distributed action-map
3. Represent in terms of behaviors implemented as control policies
4. Sequence within action-map
5. Perform autonomously
6. Behaviors can be superpositioned
7. Behaviors can be composited.

Figure 7 illustrates the teleoperational procedure we used in our research experiments.

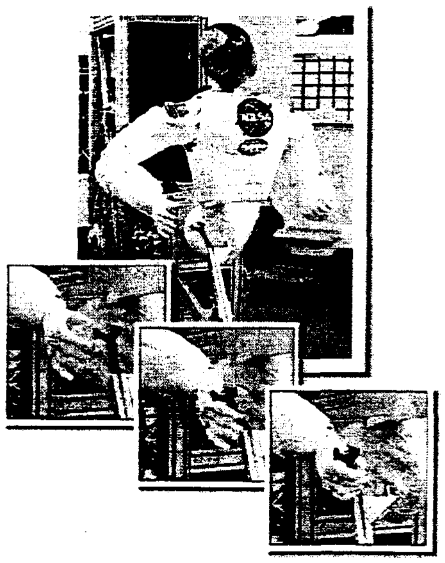


Figure 7. Teleoperational procedure using NASA-JSC's Robonaut.

Given the dynamics of Robonaut under teleoperation – its maximum velocity is limited – the durations of the episodes are relatively long and the sampling rate well exceeds the Nyquist limit. Thus the salient sensory-motor characteristics are well represented in all the trials at each of the locations and time warping for episode normalization preserves those characteristics. This would not necessarily be the case if the sampling rate were near the Nyquist limit and some of the episodes were of short duration.

3.2.4. Superposition using Verbs and Adverbs

After the resampling and averaging of the sensory-motor data from the example tasks, the data were analyzed to characterize the motions that would enable Robonaut to reach toward and then grasp a vertically oriented wrench anywhere within its workspace. This was done with an interpolation method called *Verbs and Adverbs,* (VaV) developed in the computer graphics community by [Rose 1998]. The following description is an adaptation for robotics of the algorithm from that paper. Table II lists symbols used in the description. A *verb* in this implementation of the algorithm is the motion component of a task exemplar, its motion trajectory in the sensory-motor state-space. The adverb is a specific parameterization of the motion trajectory.

In [Rose 1998], several example motions were created for articulated characters. The mapping of these motions into a multidimensional adverb space defined extremal points along axes of the space. A particular adverb extremum characterized the appearance of the associated motion. To create motions that exhibited combinations of the characteristics, a location in the adverb space was selected and mapped back into the motion space. In the work described here, the adverbs are the 3D Cartesian world coordinates of the object to be grasped (the wrench). Exemplar reach-and-grasps were acquired near workspace extrema for the robot's right arm. To perform the operation at other locations in the workspace, the VaV algorithm was used to interpolate the exemplar motions. This approach permits a limited extrapolation of the data since the subspace projection can construct new trajectories that extend parametrically beyond the exemplars. Detailed mathematical description can be found in a non-published paper [http://www.vuse.vanderbilt.edu/~rap2/papers/ in Nov. 2000].

3.3 Experimental Methods and Procedures

The VaV procedure was tested in simulation and on Robonaut. Simulation tests were run on a randomized list of 269 reachable targets in a 3D grid that covered the entire workspace and extended somewhat beyond the edges defined by the original box. The test on Robonaut was performed by affixing a wrench to a jig, and placing it arbitrarily at reachable points in the workspace. Some attempt was made to cover the entire workspace, but since the goal was to prove that Robonaut could reach randomly generated targets, a systematic selection was not used. Robonaut's vision system was employed to locate the wrench in the workspace.

The major difficulty encountered in performing these experiments was Robonaut's eye-hand coordination. The actual location of the hand can vary as the encoders that measure the joint angles are turned on and off. At the time of the tests the solution to the problem was a manual calibration with three steps. First, the arm was reset (by eye) to its zero position and the encoders were reset so that they would report zero at that location. Second, the reported point-of-reference (POR) on Robonaut's hand was changed from the standard location for teleoperation, which is on the back the hand. That location on the robot corresponds to the location of the position sensor on the teleoperator's data glove. The POR was changed to the standard location for autonomous operation, which is in the middle of the palm. Third, a wrench was placed in the workspace and was reached for manually by moving the individual joints to the correct location, then the reference location for the hand was changed again by a few centimeters. This was made as a final adjustment between the location reported by vision and that reported by the arm kinematics. After this adjustment, when the hand was grasping the wrench, the location of the hand as reported by Robonaut matched the location of the wrench as reported by the vision system.

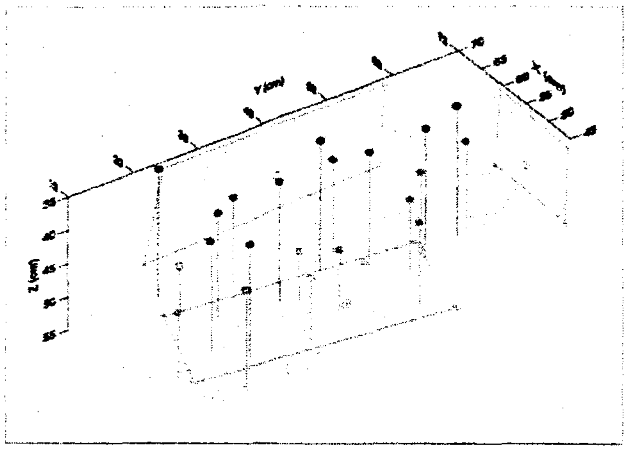


Figure 8. Plot of the 23 test locations. A wrench was placed at each of the locations marked with a disk. Projections onto each coordinate plane of the locations of the wrench during the teleoperated trials are indicated by × and ~ marks.

During the experiment, the wrench was put in 23 different locations (Figure 8). The only input that the program had was the results of the off-line analysis and the location of the wrench reported by the visual system, which was updated in real time.

3.4 Results

The simulator was of limited value in testing the procedure since it had no direct method for judging the outcome of a grasp attempt. Nevertheless, the simulator was used since it enabled a more complete analysis of the workspace than with Robonaut, due to time-sharing constraints. To ameliorate the deficiencies of simulation, numeric criteria were created from the trials run physically on Robonaut (both the original teleoperator examples and the experimental results). The trials were sorted based on physical evidence of a good or bad grasp, and then analyzed within the two categories.

Three criteria for a good grasp in the simulated data were created. The first criterion was the most obvious. If the grasp occurred too far away from the wrench to have enveloped it, the grasp could not have been successful. Any grasp that was more than 2.6 cm from the wrench location was labeled as bad. The second and third criteria concern the approach angles. If the arm motion caused the hand to approach the wrench at the wrong angle, the hand could not grasp it because the fingers, or even the hand itself, would have had to physically pass through the wrench. To judge approach angles, a vector was created by finding the direction of motion produced in the final stages of the Reach behavior.

Finally, some of the physical grasps that were incorrect were not the fault of the superposition method but of the calibration of the vision system (which was beyond the authors' control). Also, occasional inaccuracies in depth perception within various regions of the workspace resulted in errors in reported wrench location. When that happened, the hand grasped in front of, or behind the wrench. Nevertheless it did grasp at the location indicated. Since these errors were not in the superposition method itself, the corresponding grasps were defined as "marginal" and were classified as good grasps for the purpose of creating the simulator criteria and calculating results. In particular, it should be straightforward for the robot to learn that is has been unsuccessful so that it can retry the task. Although the joint trajectories are similar in both cases, there is a clearly discernable difference in the force signatures. If such a difference is consistent (we found it to be so in the 20 successful and three unsuccessful experiments) it can be detected and so used.

The Verbs and Adverbs method clearly outperformed the other two programs. It had better than 99% accuracy in the simulator trials, which were designed to cover the entire workspace. While not performing perfectly in the physical trials, it outperformed other methods used for the task.

4. Robonaut Fault Diagnostic Work

Like most humanoid robots, Robonaut is a complex electromechanical system comprised of many sensors and actuators. Overall task performance of such a system depends greatly on the proper functioning of its components. Despite the best design and maintenance practices, it is unlikely that such a complex system will be immune to random system faults. Any fault that may occur in the system may adversely affect the humanoid. Therefore it is necessary to detect, isolate, and if possible accommodate these faults as soon as they arise.

In order to perform the above-mentioned task, we plan to design an automated monitoring system for Robonaut that is expected to provide a real-time status of each sensor and actuator, and to analyze of the possible task failures given a knowledge base of system faults.

We have developed an intelligent system health monitoring technique based on a nonlinear model-based observer and a fuzzy logic framework to detect faults and identify the fault source in the robot. This framework was then developed into a hierarchical system health monitoring (SHM) technique for Robonaut where Robonaut's lower level controller publishes sensor information through NDDS, network middleware, and Self Agent analyzes the data to monitor the status of the robot. In what follows, we present a description of SHM work.

4.1 System Health Monitoring

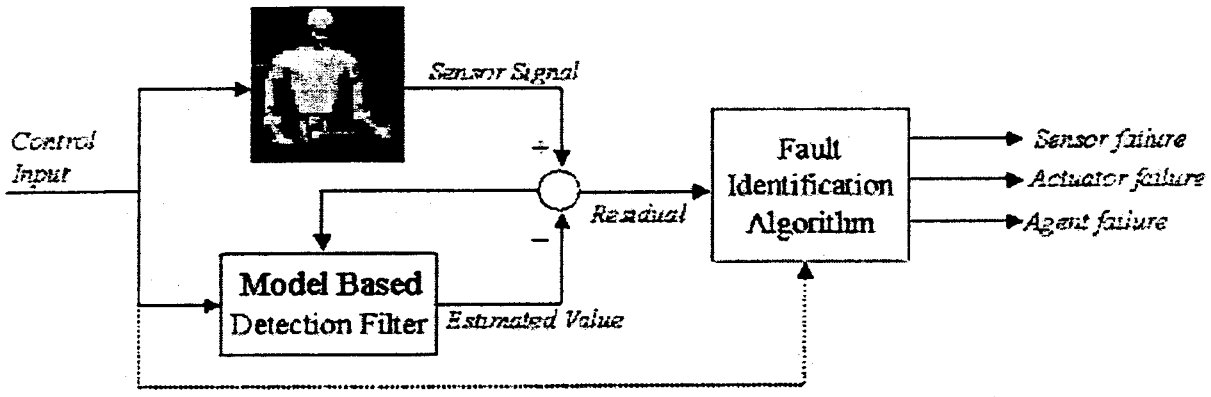
The Robonaut control system design philosophy is inspired by the human brain anatomy. The human brain embeds some functions, such as gaits, reactive reflexes and sensing, at a very low level, in the spinal cord or nerves [Ambrose 2004]. Higher brain functions, such as cognition and planning take place in other parts of the brain, including the cerebral cortex and cerebellum. Within the Robonaut control system, the very low-level functions are referred to as the brainstem. The brainstem contains the motion controllers for the 49 DOFs, sensing, and low-level sequences. The lowest-level System Health Monitoring (SHM) is designed to handle any abnormality in this level. The brainstem approach permits higher-level cognitive functions to operate independently of the low-level functions. This allows the Robonaut system to implement a variety of control methods ranging from teleoperation to full autonomy with the brainstem unaware of which higher-level control system is being used. In the human brain, this is called cognitive control [Miller 2003].

4.1.1 Low level sensor monitoring

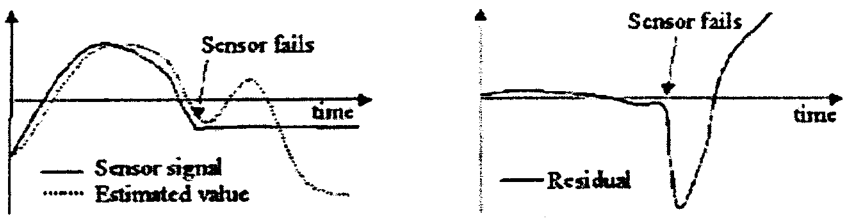
The fundamental component on which the SHM relies is its ability to monitor signals to detect faults. There are several possible component faults that can occur within Robonaut which are summarized as follows:

- Power [power supply; failure; lost connection]
- Actuator [transmission failure; encoder fault; camera fault]
- Sensor [proximity sensor fault]

Figure 9 illustrates a system block diagram and typical plots of sensor signal and residual using a nonlinear model based error residual generator. The SHM concept with a modification is being applied to the Robonaut at NASA-JSC.



(a) Block diagram of fault identification using Beard-Jones detection filter



(b) Typical plots of sensor signal and residual

Figure 9. System block diagram and plots of sensor signals

4.2 Robonaut Fault Detection Work

4.2.1 Model for the right arm of Robonaut

The right arm of Robonaut consists of seven joints as shown in Figure 10.

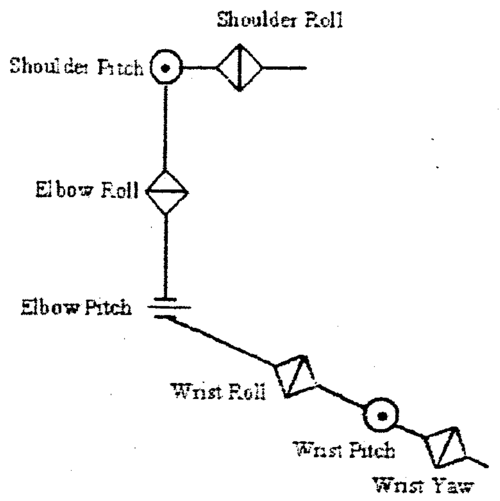


Figure 10. Joints in Robonaut's right arm.

Each joint has a servo control loop that is implemented on the real-time OS, VxWorks. In order to derive the transfer function between the motor command and the joint velocity, we made the following assumptions based on observations and heuristic knowledge:

- Transfer function between the motor command and the joint velocity is assumed to have a constant gain.
- Gravity and Coulomb friction are the dominant external forces.
- Stick-slip friction force is considered.
- Gravity is regarded as a constant because the range of joint motion was relatively small, though it is a function of joint position.
- Assume no external control forces.

Based on the above assumptions, the following structure of the model was chosen:

$$\omega = K(u - F_g - F_c sign(\omega) - F_{ss}(\omega)), \qquad \textbf{(1)}$$

where ω, u, F_g and F_c are joint velocity, motor command, gravity and Coulomb friction respectively. $F_{ss}(\omega)$ is the stick-slip friction.

Using the experimental data, we estimated the parameters of the model given by Equation (1) using a least square error method. Figure 6 shows the block diagram of the whole control system with the estimated parameters. As shown in the figure, the controller consists of an outer position control loop with a position gain k_P and an inner velocity control loop with a velocity gain k_V, a feed-forward gain k_F and a gravity compensation term F_{g_comp}. The motor command, *u*, is expressed in the following form.

$$u = k_V k_P \left(\theta_d - \theta\right) + k_V \left(\dot{\theta}_d - \omega\right) + k_F \dot{\theta}_d - F_{g\_comp}. \qquad \textbf{(2)}$$

Note that equation (2) has the same form as a PD controller except the addition of the feed-forward and gravity compensation terms.

The developed model was used to find a position error residual. To emulate encoder fault in experiments, the encoder value was set to a constant value on the control level in Robonaut's real-time control architecture. When the encoder value was set constant, the motor command increased due to the control action in Equation (2) causing the error residual to increase and hence we had to quickly stop Robonaut. One advantage to our fault detection system is that the stopping action would be automated as well as respond faster than a human who was observing and anticipating the fault, much less a human who was not paying close attention to the humanoid.

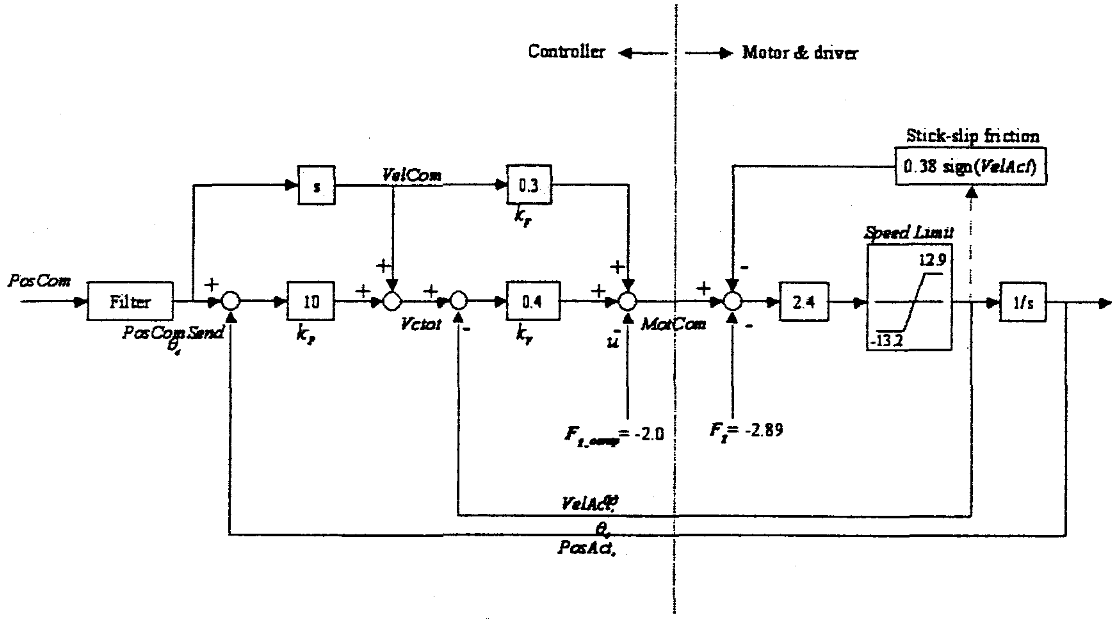


Figure 11. Block diagram for controller and motor in the Elbow Pitch joint.

4.3 Fuzzy Logic for Fault Identification

We examined the possible sources for faults and decided to begin our focus on the following:

- Sensor failure
  - There are many kinds of sensors including encoders in joints, force/torque sensors in wrist and shoulder, touch sensors in hands, and cameras. As a basic step, we concentrated on detection of encoder fault because of its importance.
  - In simulation, the motor current should increase and the error residual should increase.
  - The derivative of the sensor output should be zero.

- Actuator failure
  - This occurs when the actuator breaks and the joint goes limp. Because the motor driver supports motor health monitoring signal for malfunction of the motor, we can use this signal for fault isolation.
  - In simulation, the motor current should be near zero (implies low torque.)
  - The derivative of the sensor output should be non-zero.

We observed the experimental data to build a fuzzy logic analyzer for fault identification. Triangular membership functions were defined for all inputs to the fuzzy logic rules:

- The **Error Residual** (Encoder Reading –Estimated Model Value) could be Large Positive (LP), Positive (P), Near zero (P), Negative (N), or Large Negative (LN)
- The **Encoder Value** could be Positive (P), Negative (N) or Near Zero (NZ).
- The **Estimated Model** Value could be Positive (P), Negative (N), or Near Zero (NZ).
- The **Motor Current** could be Positive (P), Negative (N), or Near Zero (NZ).
- The **Derivative of the Encoder Value** could be Positive (P), Negative (N) or Near Zero (NZ).
- The **Derivative of the Estimated Model Value** could be Positive (P), Negative (N) or Near Zero (NZ).

The Fuzzy Logic Output Membership functions are

- **Healthy Output** (0)
- **Sensor Failure** (2)
- **Actuator Failure** (4)

Figure 12 illustrates a sample simulation screenshot for one joint.

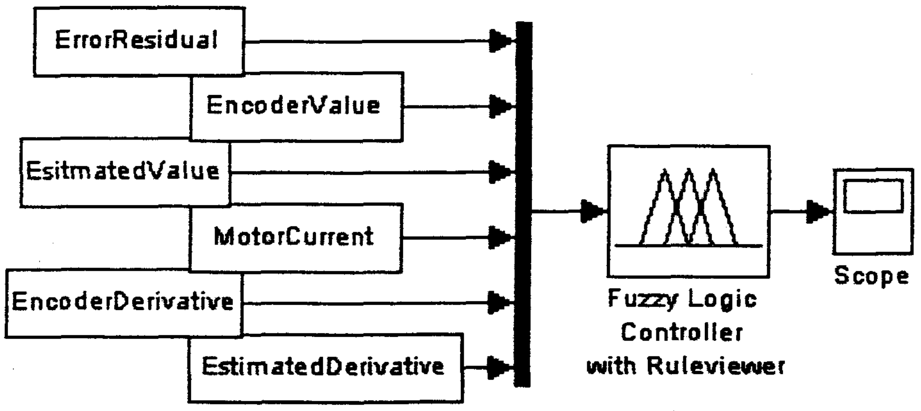


Figure 12. Sample simulation screenshot for one joint.

To identify a difference between an actuator fault and a sensor fault, the most important factors are if the residual is either no longer zero (either LP, LN, P or N), if the Motor Current acts almost normal or if it decreases dramatically (as one may expect if the actuator breaks and the motor spins), and also if the velocity of the sensor output immediately goes to zero, or if there is some oscillation.

Here are a few sample rules:

- If Residual is NZ, AND Desired Velocity is N, AND Actual Velocity is N, THEN output is Healthy Signal (0)
- If Residual is LP, AND Desired Velocity is N, AND Actual Velocity is NZ, AND Motor Current is P, THEN output is Sensor Failure (2)
- If Residual is LN, AND Desired Velocity is P, AND Actual Velocity is NZ, AND Motor Current is NZ, THEN output is Actuator failure (4)

4.4. Simulation and Experiment

Using data gathered from the Robonaut API, we calculated the desired encoder value using the derived dynamic model. Figure 13 shows at 6.5 seconds, the encoder signal was kept at a constant value of -90°, which emulates an encoder fault.

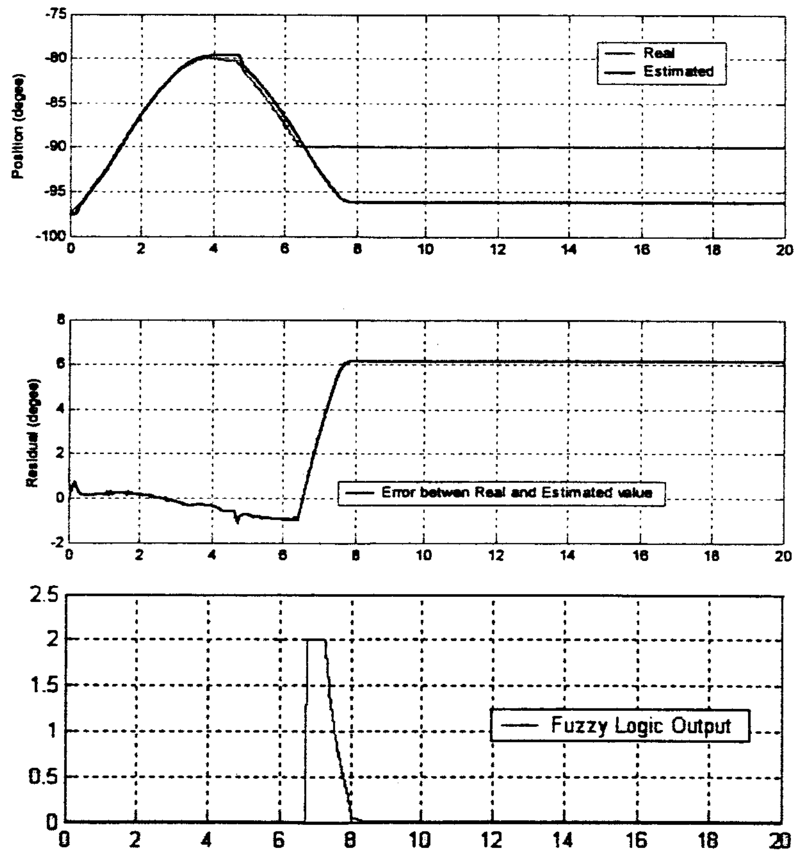


Figure 13. Encoder fault detection: (a) Actual and estimated encoder value;
(b) Difference between actual and estimated encoder value;
(c) Output from fuzzy logic based fault identification block.

Figure 13 also shows that the error between the estimated value and the real encoder value increased after the occurrence of the encoder fault. Therefore, we could easily detect the encoder failure by observing the residual. The third diagram shows the output of the fuzzy logic rules. While the encoder reading matches well with the model based estimator, the output of the Fuzzy Logic is 0 (for a healthy signal). Once the fault is introduced, the Fuzzy Logic outputs a value of

2, which corresponds to an Encoder Fault. At 8 seconds, the emergency stop button was pressed causing many sensor values to stop. As shown in the error residual plot, when everything is functioning correctly, the residual is not exactly zero. However, using Fuzzy Logic to set tolerances and to examine other important information, we can know that everything is functioning properly.

5. IMA, Behavior Generation, and Learning

5.1 Intelligent Machine Architecture

A humanoid robot is an example of a machine that requires intelligent behavior to act with generality in its environment. Especially in interactions with humans, the robot must be able to adapt its behaviors to accomplish goals safely. As grows the complexity of interaction, so grows the complexity of the software necessary to process sensory information and to control action purposefully. The development and maintenance of complex or large-scale software systems can benefit from domain-specific guidelines that promote code reuse and integration. The Intelligent Machine Architecture (IMA) was designed to provide such guidelines in the domain of robot control [Kawamura 1986] [Pack 1998]. It is currently used to control ISAC and a set of mobile robot [Koku 2003] [Kawamura 2002].

IMA consists of a set of design criteria and software tools for Windows NT/2000 that supports the development of software objects that we call "agents". An *agent* is designed to encapsulate all aspects of a single element (logical or physical) of a robot control system. A single hardware component, computational task, or data set is represented by an agent if that resource is to be shared or if access to the resource requires arbitration. Agents communicate through message passing using DCOM, the Distributed Component Object Model service of Windows NT/2000. IMA facilitates coarse-grained parallel processing because of the loose coupling afforded by message passing and because DCOM allows software objects on separate computers to be treated as if they were local to each other. Each agent acts locally based on its internal state and provides a set of services to other agents through various relationships. The resulting asynchronous, parallel operation of decision-making agents simplifies the system model at a high level. IMA has sufficient generality to permit the simultaneous deployment of multiple control architectures. A behavior can be designed using any control strategy that most simplifies its implementation. For example, a simple pick and place operation may be most easily implemented using a standard Sense-Plan-Act approach, whereas visual saccade is more suited to subsumption, and object avoidance to motion schema.

There is a two-level hierarchy of IMA agents comprising *atomic agents* and *compound agents*. A compound agent contains or depends on other agents for its primary function. An atomic agent encapsulates a single resource; it neither subsumes nor references any other agent. (The term "atomic" is used in the sense of the Greek word *atomos*, literally "indivisible".) Each controllable hardware device or common data resource on the robot has an associated *hardware/resource* atomic agent. A dataset or computational procedure associated with a specific object in the robot's external environment has an associated *environment* atomic agent.

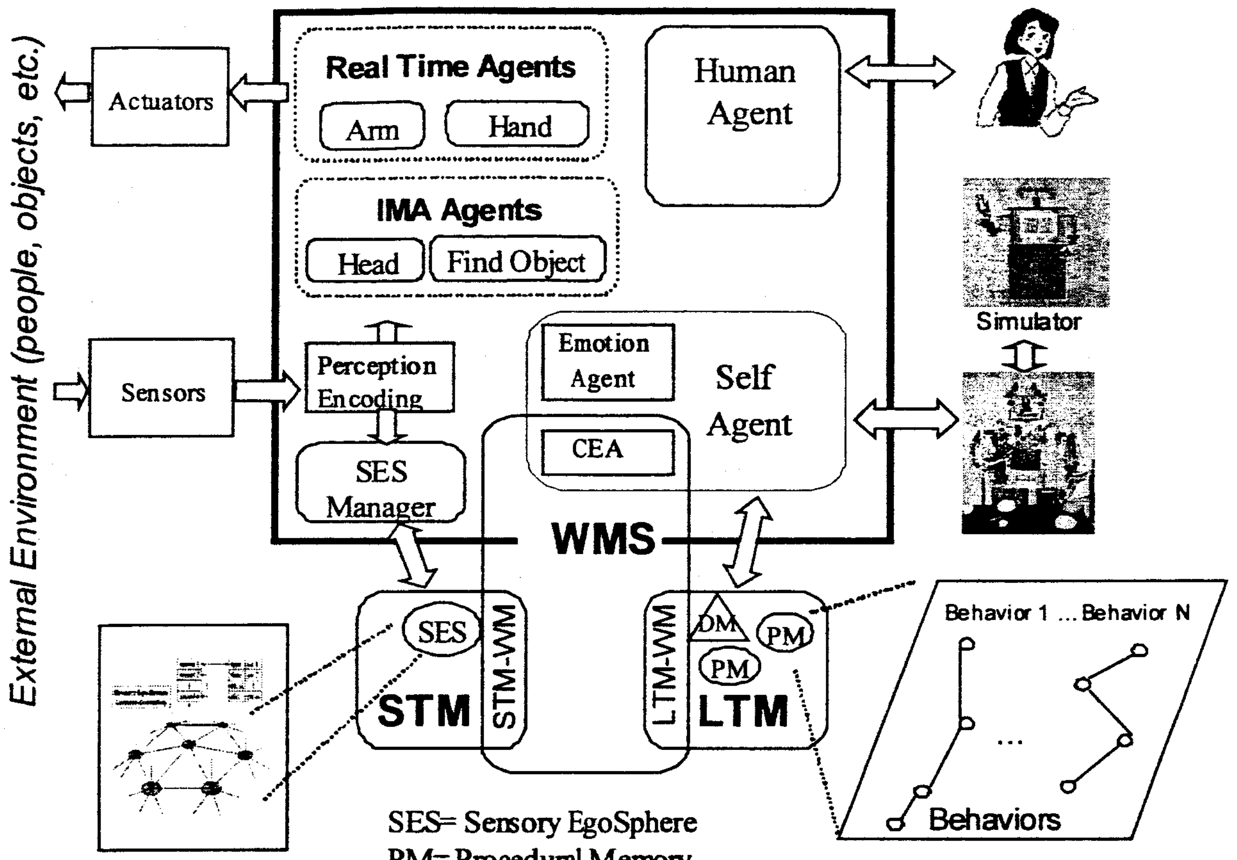
Within IMA, any existing agent can be accessed by any other agent. Their connectivity, as defined by message passing, is flat -- without hierarchy. However, a virtual or logical hierarchy is implied by the structure of a compound agent. Various levels of abstraction form within the control system as needed but are not fixed.

IMA works very well to promote software reuse and dynamic reconfiguration. However, the large systems built with it have experienced scalability problems on two fronts. First, as the system exceeds a certain level of complexity it is difficult for any programmer to predict the interactions that could occur between agents during actual operation. This level seems to be higher than for a direct, sequential program. But that level has been reached in the development of ISAC. The other scalability problem may or may not be a problem with IMA itself but may be an inevitable consequence of increasing complexity in a system based on message passing. The asynchronous nature of message passing over communications channels with finite bandwidth leads to system "lock-ups". These occur with a frequency that apparently depends on the number of agents in the system. It may be possible to minimize this problem through the use of system-self monitoring or through a process of automatic macro formation. For example, the system could, through a statistical analysis, recognize the logical hierarchies of agents that form repeatedly within certain tasks or under certain environmental conditions. A structure so discerned could be used to "spin off" copies of the participating agents. These could be encapsulated into a macro, a compound agent that optimizes the execution and inter-process communications of the agents involved. For such an approach to be most useful, it should be automatic and subject to modification over time frames that encompass several executions of a macro.

5. 2 ISAC Memory Structure

Within IMA, the robot itself is abstracted as a *self agent (SA).* The self agent uses a memory database structure, consisting of short-term and long-term structures, to determine the appropriate situational control commands for the robot. The short-term memory (STM) uses the Sensory EgoSphere (SES) to represent short-term external events in the environment The SES is a data structure that provides a short-term-memory to store events, such as the state of external human agents. Long-term memory (LTM) contains information about procedures considered intrinsic to the robot. The self-agent uses both the short and long term memory structures to provide control. The STM is used by the self agent to provide an estimate of the current external state for determining appropriate task-level intentions for the robot. Based on these intentions, the self agent uses procedures in the LTM to provide control commands to accomplish the robot's intentions. In this paper, we propose a self-agent that uses derived behaviors as procedures to produce motion control for achieving robot objectives. Within the self-agent, a *central executive controller (CEC)* uses the derived behaviors in the LTM.

A more recent addition to the cognitive architecture is that of a Working Memory System (WMS). This system will link the memory structures with the Central Executive Controller (CEC), selecting only those structures appropriate for the task. Currently we have integrated the WMS with the derived behaviors in the LTM to produce movements for ISAC. In the future, we will integrate the system fully with the STM as well as the more executive functions of the CEC.



SES= Sensory EgoSphere
PM= Procedural Memory
DM=Declarative Memory
CEA=Central Executive Agent

Figure 14. Component view of the IMA multi-agent-based system.

5.2.1 Automated behavior derivation

To utilize behavior-based control properly, a substrate of behaviors is needed that can express the desired range of the robot's capabilities. In deriving such skill capabilities for a robot, we assume that captured human motion is structured by underlying behaviors and that the performed activities are representative of the robot's desired capabilities. We further assume that each underlying behavior produces motion with a common spatial signature and is typically performed in sequence with common preceding and subsequent behaviors. Given these assumptions, we can automatically derive a behavior vocabulary using a spatio-temporal extension of Isomap. Such derived behavior vocabularies are structurally similar to Verbs and Adverbs vocabularies discussed previously (Section 3) in that each behavior is defined by a set of exemplar motions that are generalized through interpolation.

The behavior derivation method consists of four main components. The derivation system takes as input a single continuous kinematic motion as a time-series of joint angle values. This motion is segmented into intervals based on some heuristic defining separating events, with each segment assumed to be an atomic motion. Several methods exist for segmenting time-series data. We use Kinematic Centroid Segmentation which treats each limb as a pendulum and greedily seeks "swing" boundaries. Segmentation with time normalization produces an ordered set of data points in a D*N dimensional space, where D is the number of DOFs and N is the number of frames in each motion segment. Spatio-temporal Isomap works by finding *common temporal neighbors (CTN),* pairs of points with small spatial distances whose sequential neighbors also have small spatial distances. Through transitivity, CTNs for *CTN connected components* that result in easily distinguishable clusters in the produced embedding. Furthermore, the number of clusters is found automatically using no *a priori* cardinality estimate. Each cluster, called a *primitive feature group*, is a set of exemplars with a common spatio-temporal theme that is indicative of some underlying behavior. Interpolation is used within a cluster's set of exemplars to sample new motions from the underlying *primitive behavior*. By densely sampling cluster exemplars, each primitive behavior is uncovered as a set of motion trajectories that form a low-dimensional flow field manifold in joint angle space. Additionally, further embedding/clustering iterations can be applied to successive embeddings for clustering higher-level *meta-level behaviors* as sequential transition models of primitive behaviors. Derivation process results are a behavior vocabulary consisting of primitive behaviors that represent a family of kinematic motion across a span of variations, and meta-level behaviors, which represent sequential combinations of the primitives and index into them to produce motion [Matarić 1992].

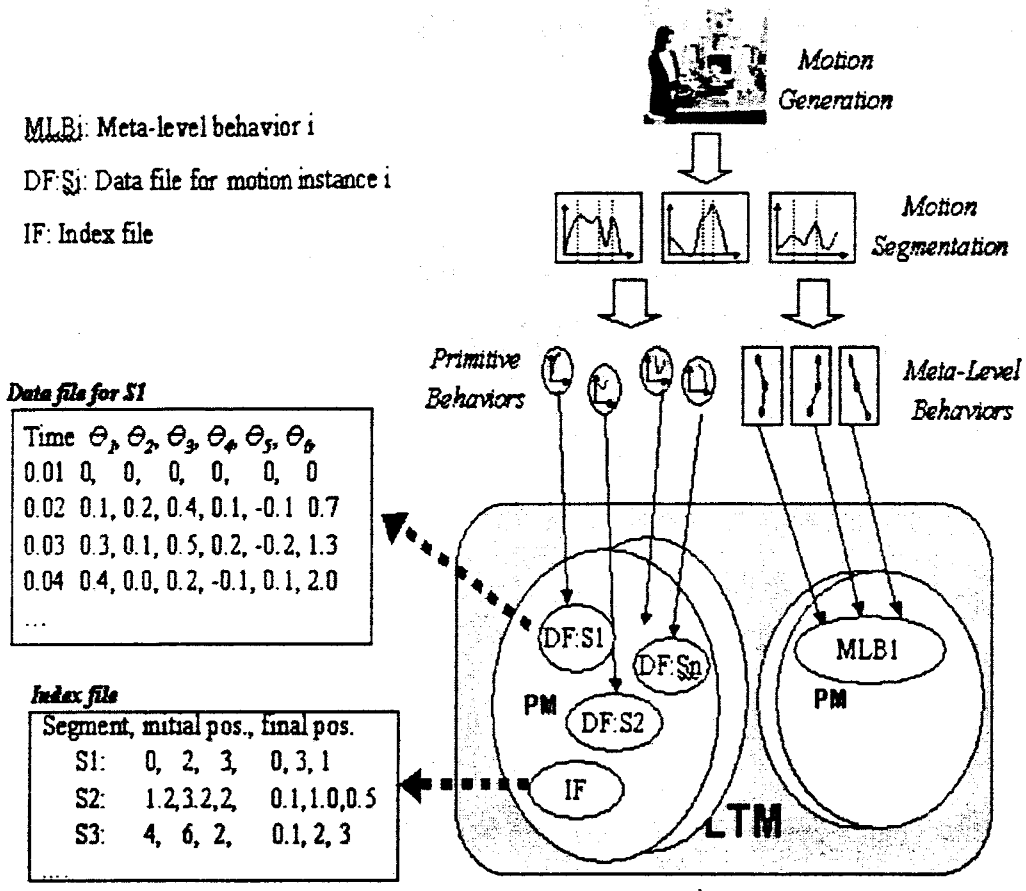
5.2.2 Role of behaviors in Procedural Memory

In our approach, the derived vocabulary is assumed to be an intrinsic substrate of basic robot skills. Consequently, this vocabulary is stored as long-term memory, more specifically as Procedural Memory (PM) (Figure 15). Generally, PM is a memory unit for storing a skill and procedure and is involved in tasks such as remembering how to reach to a point [Carter 2000].

Each behavior in the vocabulary is stored as a PM unit. Each primitive behavior is stored as a set of trajectories in joint angle space with an indexing structure stored as a PM unit. This indexing structure stores the initial and final Cartesian coordinates for all arm trajectories in a primitive behavior.

The Central Executive Controller (CEC) uses the PM index to search the desired initial and final position of motion. It uses the PM units to translate robot goals into control commands for accomplishing the goal by searching for PM units suitable for accomplishing goals and then uses the indexing structure of the PM units to produce the desired motion.

For example, if the robot has the goal of *"reach-to XYZ"*, the CEC can determine that PM *"reach"* at coordinates *"XYZ"* will accomplish this goal. In a sense, the goal *"reach-to XYZ"* spawns the intention to *"reach, XYZ"*, and this intention directly specifies which action to take.



Data file for S1

```
Time θ1, θ2, θ3, θ4, θ5, θ6
0.01 0,   0,   0,   0,    0,   0
0.02 0.1, 0.2, 0.4, 0.1, -0.1  0.7
0.03 0.3, 0.1, 0.5, 0.2, -0.2, 1.3
0.04 0.4, 0.0, 0.2, -0.1, 0.1, 2.0
...
```

Index file

```
Segment, initial pos., final pos.
   S1:     0,  2,  3,    0, 3, 1
   S2:     1.2,3.2,2,    0.1,1.0,0.5
   S3:     4,  6,  2,    0.1, 2, 3
....
```

Figure 15. Structure of the LTM database

5.2.3 Generating Motion from Stored Behaviors

ISAC can react to the environmental changes autonomously by generating desired motions from the stored behaviors. The behaviors stored in the PM are managed by a central planner, which performs searching across behaviors and within each behavior. Generating new motions from the generic behaviors involves a planning algorithm and an interpolation method.

The search mechanism in the CEC receives an estimate of the current external state of the environment from the Sensory EgoSphere (SES) to determine appropriate task-level intentions for the robot. Based on these, CEC uses two-step tasks to search the LTM to provide control commands to accomplish the robot's intentions (Figure 16). As mentioned above, PM units store primitive behavior motion data as a dense sampling of motion trajectories representing the span of variations of a behavior. In the case of a match, the motion trajectory is sent to the CEC. If there is no match, a new motion is interpolated using the local neighbor motions from the primitive behavior database. Shepard's interpolation [Shepard 1968] is used. The generated motion data are sent back to the planner. The actions are then sequenced and sent to the hardware controller(s) to accomplish the robot's intentions.

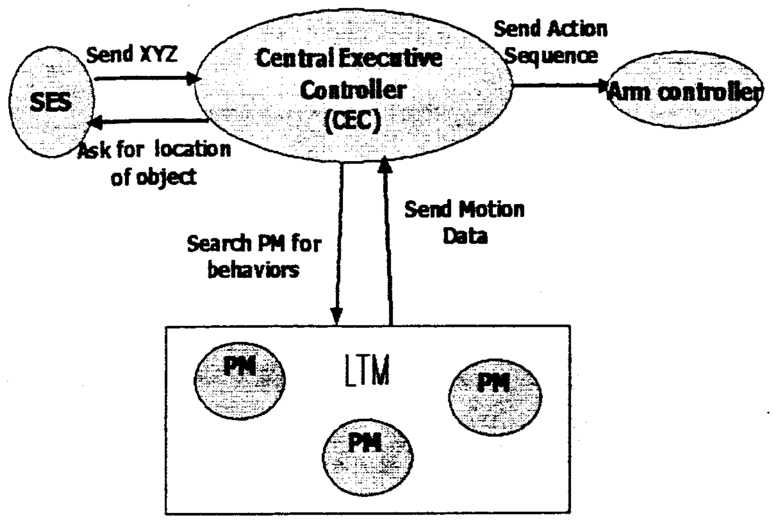


Figure 16. Structure of the search mechanism for generating new motions from stored behaviors.

5.3 Experimental Results

ISAC was driven by a human operator to reach to 5 random points (A, B, C, D, E) on a table, a limited working space, as shown in Figure 17.

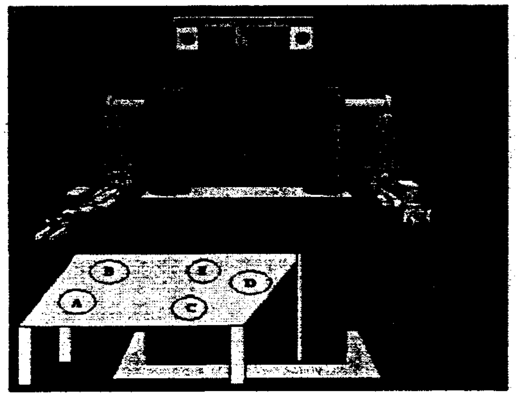


Figure 17. ISAC is taught to reach 5 points on a table.

The motion data stream consisted of 1241 frames, which contained only one reaching behavior. The stream was segmented using the Kinematic Centroid Method resulting in 10 segments. Each segment was normalized to 100 frames. The parameters for the K-nearest neighborhood, common-temporal (c_{ctn}) and adjacent temporal neighborhood (c_{atn}) constants for spatio-temporal Isomap were selected (*). The structure was embedded to 3 dimensions. The segments that were similar came closer after applying spatio-temporal Isomap (Figure 17). Each segment in the embedding is numbered). The primitive feature groups were produced through clustering. The closer segments were clustered in the embedded space within 0.1 of the diagonal of the embedding bounding box to derive the primitive behaviors. The clustering resulted in two derived primitive behaviors: reaching and returning to home positions.

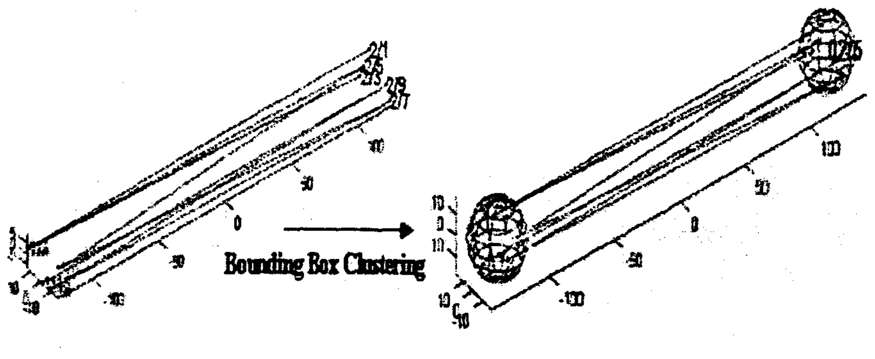


Figure 18. Embedding and clustering of first motion stream using spatio-temporal Isomap. Lines show temporally adjacent segments.

100 random coordinates with the bounding box method were interpolated to obtain new joint space trajectories for each primitive behavior [Shepard 1968]. The reaching behavior and return to home behavior are interpolated (Figure 19).

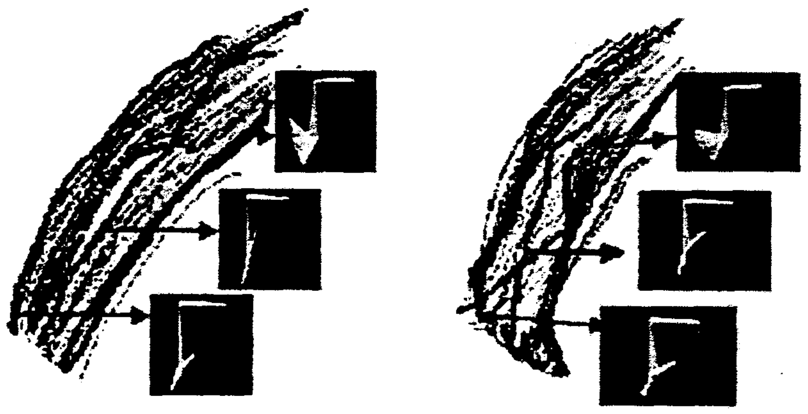


Figure 19. Results from interpolating selected action units. Each plot shows trajectories in Cartesian coordinates. (Left) reaching, (Right) returning to home.

Spatio-temporal Isomap was applied a second time. Sweep-and-prune clustering was then used in the embedded space with a 0.1 threshold value to derive the meta-level behaviors (Figure 21). The segments from the same primitive feature groups became proximal after the first application of Spatio-temporal Isomap. After the second application of spatio-temporal, the primitives typically performed in sequence were collapsed (Figure 22). Thus segments were collapsed from the corresponding primitive feature groups in the sequence into the higher-level behaviors. The meta-level embedding resulted in the same number of clusters as in the primitive embedding. This convergence of the embeddings indicates there is no higher-level structure in the motion greater than the primitive level. Thus, we consider the vocabulary has one meta-level behavior that encompasses the two primitive behaviors.

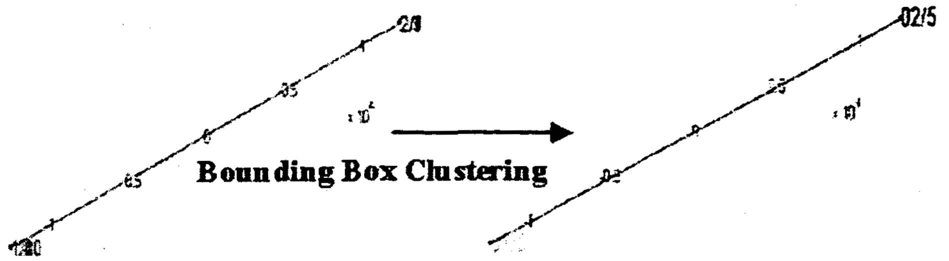


Figure 20. Embedding and clustering of second application of Spatio-temporal Isomap. Proximal motion segments are grouped to form behaviors.

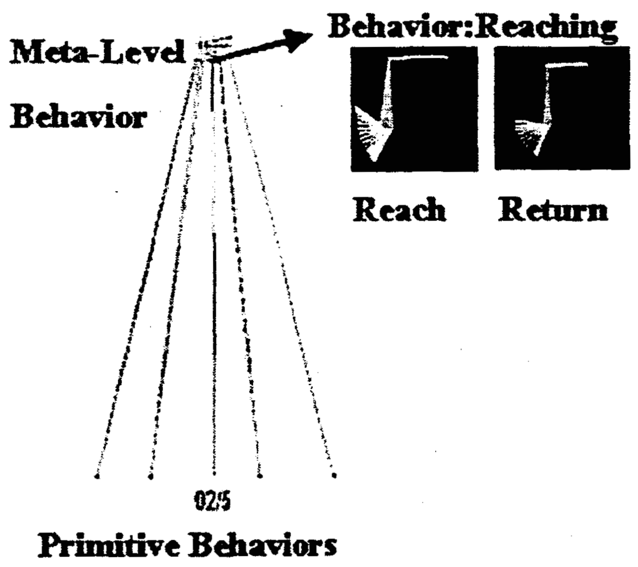


Figure 21. The transitions between the segments that derive meta-level behavior for each reaching motion. Lines indicate transitions between actions.

6. Cognitive Control and the Central Executive Agent

6.1 Cognitive Control in Human and Robot

Cognitive control in human is the ability to consciously manipulate thoughts and behaviors using attention to deal with conflicting goals and demands (Botvinick, et al, MacLeod and Sheehan). As levels of human behavioral processes range from reactive to full deliberation, cognitive control must be able to switch between these levels to cope with the demand of task and performance, particularly in novel situations. Cognitive robots like Robonaut and ISAC, thus, should have cognitive control ability to handle unexpected situations during routine operations. Toward this goal, we are implementing an adaptive control process called *modular control* for ISAC. The process makes use of a working memory system to allow ISAC to reason about task context in the error driven manner during its arm manipulation.

6.2 ISAC Control Architecture

A humanoid robot like ISAC is an example of a machine that requires intelligent behavior to act with generality in its environment. Especially in cooperation with humans, the robot must be able to adapt its behaviors to accomplish tasks under conflicting demands. At Vanderbilt, we adopted a design strategy which allows us to control ISAC's behavior robustly similar to human's cognitive control function. We hypothesize that by using modular control and a working memory system, ISAC will efficiently maintain its attention and task-related information chunks to adjust its actions accordingly.

During the original MARS grant, Intelligent Machine Architecture (IMA) was designed to control ISAC behavior and communicate with humans (Kawamura). Since then, various new IMA agents and memories have been added such as the Central Executive Agent (CEA) and the Working Memory System (WMS). Figure 21 depicts the current IMA-based ISAC control architecture and the memory structure.

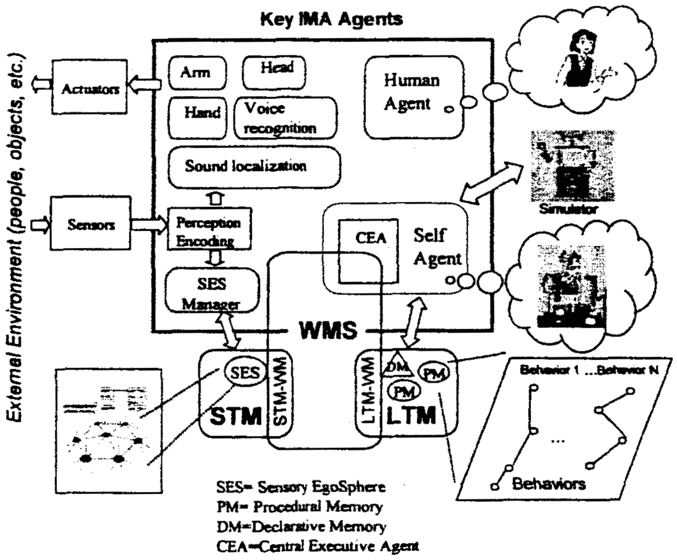


Figure 22. Key IMA agents and the memory structure.

6.3 ISAC Memory Structure

Sensory processing agents write data to the Sensory EgoSphere (SES) which acts as a short-term memory (STM). The STM thus maintains a short-term spatio-temporal relations of various objects in the environment. Long-term memory (LTM) stores motor skills and semantic knowledge. Motor skills are represented as primitive motions (PMs) which are chained together to generate behaviors. Semantic knowledge is stored in another part of LTM called declarative memory (DM). We are now adding a third memory structure called the working memory system (WMS) to facilitate behavior learning and task execution.

The WMS is expected to interface closely with the Self Agent, the STM and the LTM to achieve the following goals:

1. Selection of task-relevant primitive motions
2. Generation and execution of behaviors
3. Task execution and learning.

Activated elements of STM and LTM are shown as STM-WM and LTM-WM [Franklin 2003]. In addition to the WMS, we are developing a Central Executive Agent (CEA) to handle the behavior generation and execution aspect of cognitive control. The agent is a part of the Self Agent and interfaces directly with the WMS.

Research this year focused on accomplishing the second goal: the generation and execution of behaviors. This has been accomplished through the use of TD-Learning. Simple efforts have also been made towards the first and third goals. The next section describes the details.

6.4 Central Executive Agent

One of the most important agents in the ISAC architecture is a cognitive agent called the Self Agent (SA). Being developed to represents the robot's "self," the SA currently maintains information about the robot's internal state and the progress of task execution. The internal representation of the robot's self -model is expected to be continuously updated. Human cognitive abilities such as past experience and emotion will be adapted to be a part of the SA in the future.

Recent cognitive psychology research supports the idea that working memory plays a very important role in human cognitive control. Cognitive control in ISAC is partially inspired by a working memory model in which the control of executive processes is done by a component called the Central Executive. In that model, the Central Executive controls two working memory systems, namely the phonological loop and the visuo-spatial sketch pad. These two systems are responsible for both the storage and processing of linguistic and visio-spatial information, respectively. Cognitive control in ISAC is enabled by a component with a similar function called the Central Executive Agent (CEA) that interfaces with a working memory system (WMS) that allows task related information to be actively maintained and analyzed in order to successfully select and execute an action for an assigned task.

Primitive motions previously taught or learned are stored in the LTM [Erol 2003]. When a new command or goal is given, the CEA selects a small number of behaviors that are believed to be the best for the current task, and loads them into the WMS based on past experience as measured within a Reinforcement Learning (RL) module. Within the WMS, the state estimator and behavior controller are assigned to each behavior. At current time t, the CEA takes the estimated state of the robot x_t and goal state x^* to determine the amount each selected behavior will contribute to a particular action based on its relevancy to the task context. The computation of task relevancy is based on the original concept of modular controllers and their responsibilities [Wolpert 1998].

For each behavior *i*, its task relevancy is computed by

$$\lambda_i = \alpha_c \frac{e^{-|ec_i|^2/\sigma_c^2}}{\sum_{j=1}^{N} e^{-|ec_j|^2/\sigma_c^2}} + \alpha_g \frac{e^{-|eg_i|^2/\sigma_g^2}}{\sum_{j=1}^{N} e^{-|eg_j|^2/\sigma_g^2}}.$$

Where, $ec_i(t) = x_t - \hat{x}_t^i$ is the current state error and $eg_i(t) = x^* - \hat{x}_t^i$ is the goal state error. The values of α are varying based on the type of task and values of σ are adjusted based on the type of the actuator. From λ_i, the CEA produces appropriate weights for the behaviors within the WMS to combine control signals from behavior controllers before the result is then sent to the robot for action execution. Figure 22 shows the interaction of components in ISAC's cognitive robot architecture with relation to modular control.

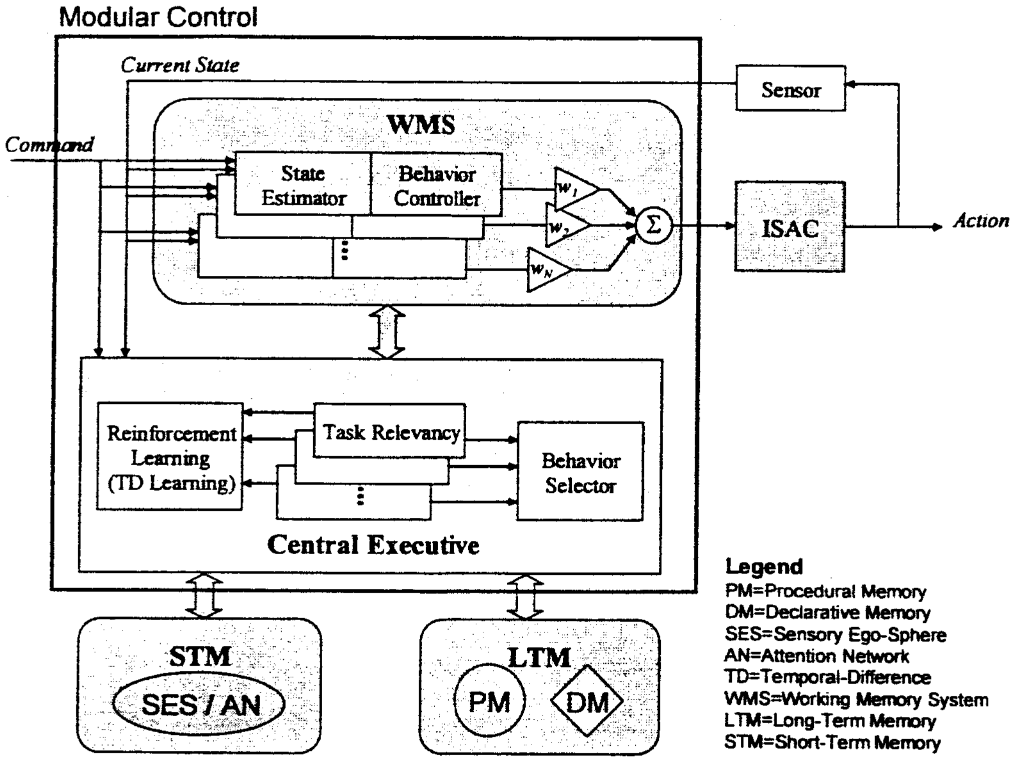


Figure 22. Modular control in cognitive robot architecture and related components.

Finally, the role of the reinforcement learning system is to learn to select appropriate behaviors for a given task. Learning is done at the end of a task execution. A new reward is calculated for each set of behaviors after attempting the task. The reward is equal to the average relevancies of behaviors within the set over the time it took to complete the task, discounted by the task completion time. Thus, the set of behaviors that complete the task most quickly with the most precision will be rewarded and are likely to be selected in the future. ISAC learns which set of behaviors to select for each task. New, similar tasks can be built on the learned values.

6.5 Experiment

To prove the concept of this modular control system, we have created an experiment in which ISAC is asked to perform a *pointing* task. In this task, multiple PMs must be combined in order for the robot to reach a point that is not accessible by the execution of any single PM. Over a small number of trials, the CEA learned to select appropriate PM units, loaded into the WMS, combined the PM units to produce the desired movement as shown in Figure 23. The combination of control signal represented by weights of a successful trial is shown in Figure 24. The system was successful in this simple task, and will be expanded in the future to handle a larger PM size as well as to execute more complex movements.

In this case, a small number of PM units were used for the task, minimizing the initial period in which the system randomly selects PM units before finding a set that returns a reward. In the future, we expect to include more information within the PM units to assist the TD-Learning unit in selecting the units. This will prevent the need to have large amounts of settling time for the system to learn a task.

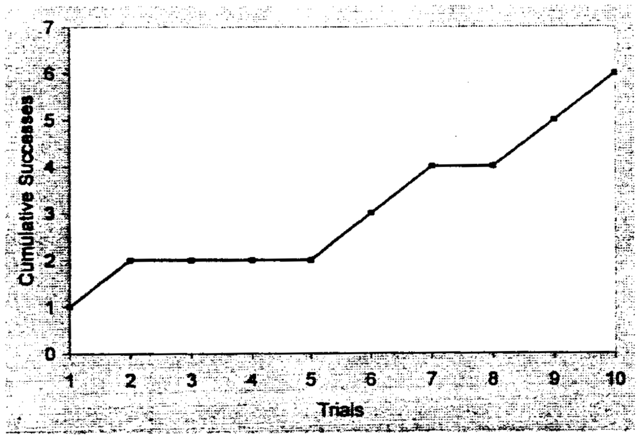


Figure 23. Cumulative successful trials.

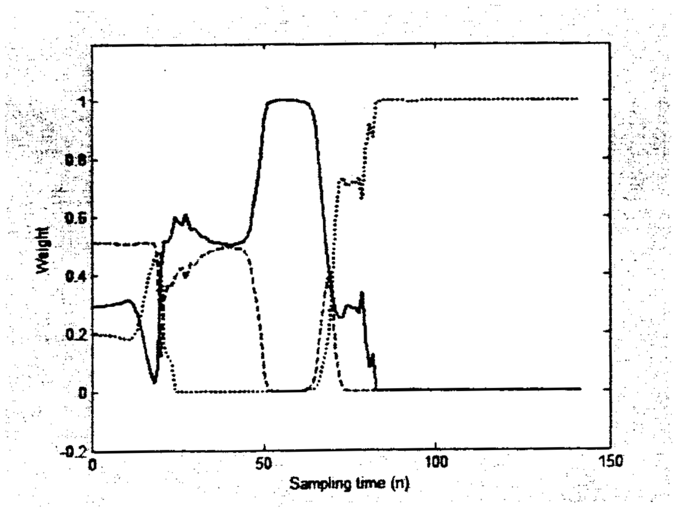


Figure 24. Weight distribution among behavior motion during a successful trial.

7. Segway RMP Research

7.1 SES-based Segway Navigation

The CIS was given a Segway RMP to work on during the time of this grant, with the intention of developing innovative application software. Our Segway RMP is shown in Figure 25. Hardware added were SICK LMS200 Laser sensor placed on top of an independent rotating platform, and a panoramic array of six cameras. The rotating platform consist of a stepper motor, controlled via a computer's parallel port, and a table to set various sensors on. The panoramic camera array consist of six cameras connected to a multiplexer which consecutively captures one image at a time from the environment and feeds that image to the computer via a standard USB port.

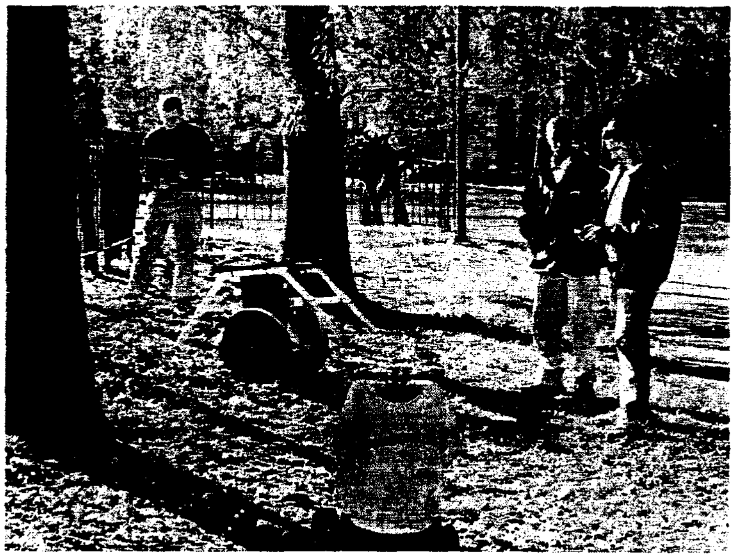


Figure 25. Conducting Segway exercises outside with Pioneer AT-2.

Software installed includes NDDS, a communications tool used to share all information traveling to and from the robot with any process or computer that requires it (with help from Bill Bluethmann and Ken Adler from NASA JSC) and Player, a robot server used primarily to act as the driver connecting the robot and sensors to the computer. We decided to extend our 2-D Egocentric navigation algorithm developed under the original DARPA grant [Kawamura 2002]. We decided to extend to 3-D SES-based navigation. Figure 26 illustrates the concept.

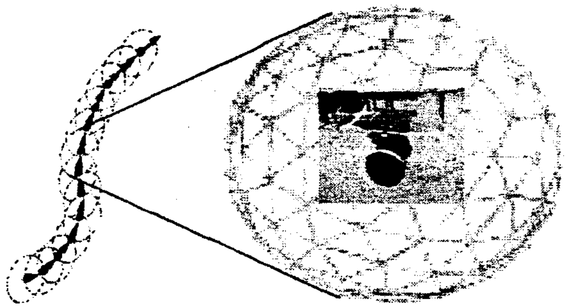


Figure 26. 3-D verion of SES with Segway.

To construct the SES for Segway navigation, we focused on utilizing camera vision and a laser scanner as separate entities, each working to identify objects of interest in the immediate environment.

Using such contrasting sensors allows us to extract a wide range of information from the environment. The work with the camera as been to obtain 360 degree images by arranging seven separate cameras in an outward circle, take a snapshot with each camera, and combine it with the other snapshots to obtain a complete panoramic view.

Using this sensor with the unique dynamics of the Segway robot has both benefits and problems. One of the main benefits is that we can extract images from the environment at near human eye-level, allowing us to draw relationships from the environment similar to those that a human would draw (it is important to note that most environments that the robot will travel in have been designed for humans seeing at human eye-level).

One of the main problems with using this system on the Segway is that while traveling, the Segway tends to sway back and forth, an effect that is only amplified the higher up one travels on the robot. Due to this, the robot must stop in order to collect an image scan. Currently, the work on this system has been to take image scans, process them to identify specific landmarks, and localize the position of the landmark relative to the image. This work will facilitate the construction of a SES on the Segway similar to that already in use on other robots in our lab.

71.1 Laser scanners and a spatial map of the environment

The work regarding the laser scanner has taken a similar approach in many aspects. In the past, laser scanners have been used to detect walls, openings (such as doorways, etc.), and obstacles in the environment.

For this project, the laser is rotated perpendicular to the axis it scans about so that it can collect 3-dimensional views of the environment. From this scan, objects of interest are found, pathways that the robot can use are calculated, and obstacles (to a degree) can be avoided. The work done thus far has been that once an obstacle has been detected, to calculate its position relative to the robot and to all of the other objects detected in the environment. By doing this allows the robot to create a spatial map of the environment based on the objects that it has seen, without prior knowledge of the identity of the objects it is looking at.

A sample laser scan, with different colors to assigned to different objects of interest, along with the corresponding spatial maps is shown in Figure 27.

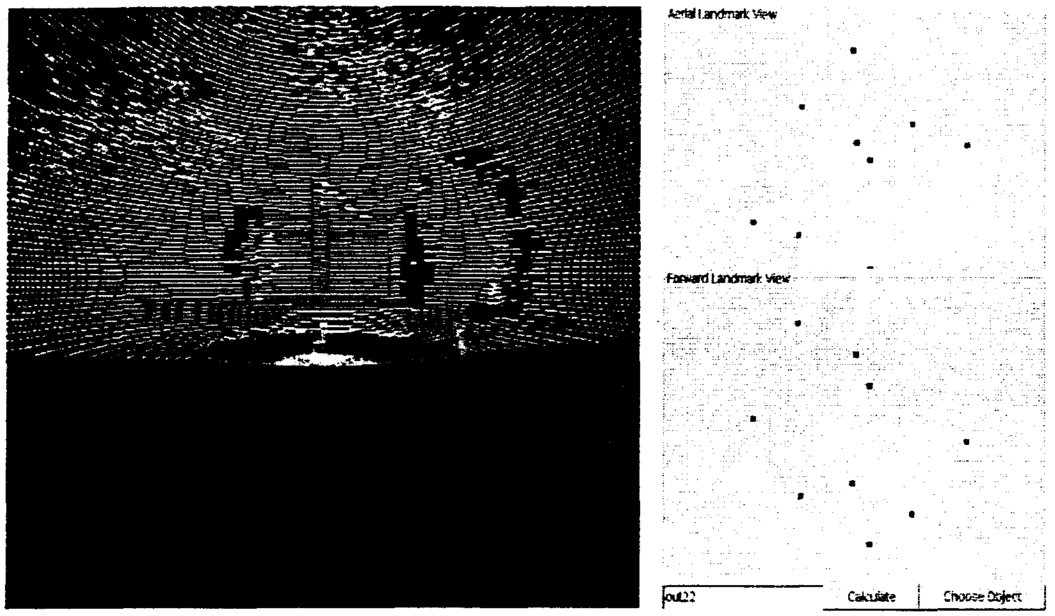


Figure 27. Sample laser scan and spatial map.

The future work for these systems is to combine the information collected from each separate system into an overall view of the environment, a SES for a mobile robot. We hypothesize that having such a SES will allow the robot to navigate environments similarly to the way humans navigate, as well as to obtain useful navigation information about new environments, which can then be shared with a human user or other robots in need of that information.

7.1.2 SES/Vision-based Segway navigation

We have been working on the original vision-based landmark recognition system called the Egocentric Navigation (ENav) algorithm (Kawamura et al. 2002) under the original MARS program. This system was designed to identify brightly colored artificial landmarks. Our current work improved on the original 2-D ENav, making it more robust, and additionally, porting it to run under Linux. Separately, we have been working on a more sophisticated vision system that we believe has the potential to learn to recognize natural landmarks.

An interactive approach to gather, analyze, and present processed visual information to the user was tried. Such an interaction enables the user to identify which landmarks the robot can reliably recognize in the environment, and then automatically design and implement detectors for these objects. The user and the robot working together will explore the visual features that provide the best performance in object recognition.

Because the features have semantic meaning, both the user and the robot have the same shared semantic context, and as a result, the user has an understanding of how the robot is identifying the objects. This knowledge can be used to refine the recognition methods, or may lead the user to decide that the current feature set is inadequate and more features must be added to the system. Such a system has significant advantages during the development process as well as supporting the training of the robot in the field as conditions change, which, in a real situation, is inevitable.

A conceptualization of the system is given as follows. First, the user teaches the robot about objects the user wants the robot to recognize, by showing the robot the various objects in a variety of images collected by the robot. Next, the robot examines these objects, computes visual features from them, and searches for discriminating features for identifying the objects. At present the search methods involve two approaches. The first is based on Maximum A Posteriori (MAP) estimation while the second approach adapts a machine learning approach (i.e., decision trees) to the automatic learning of fuzzy membership functions for the object classes.

It is important that the chosen features used by the system be such that they have a semantic meaning that can be expressed in human language. Choosing features in this fashion provides a shared semantic context for subsequent dialog between the human and the robot. Initially, we used a generic set of features that can be described as 250 different colors, two texture measures (roughness), one symmetry measure, and the presence of straight lines at 180 different angles. This yields 433 different visual features, each describable in clear language terms. This set of features was observed to work relatively well in outdoor environments, but it's color perception was too coarse for the more subtle color patterns of indoor environments. For increased color sensitivity we have now increased the number of color features to 10,000, and the observed results are significantly better for our indoor environment experiments.

The MAP approach mentioned above is a well known method, however, the fuzzy set approach deserves some extra comments. The training set is examined using decision trees, e.g., the C4.5 algorithm. The result is a tree structure with the branches of the tree being created by different values of the features. Since the features have semantic meaning, traversing the tree produces a description of the features important for identifying different objects. We have formulated these descriptions in the form of fuzzy IF-THEN rules.

Initial experiments on natural objects such as trees have also been very encouraging. Detectors for finding trees were automatically designed by the system and tested on images containing trees. The results are shown in Figure 28.

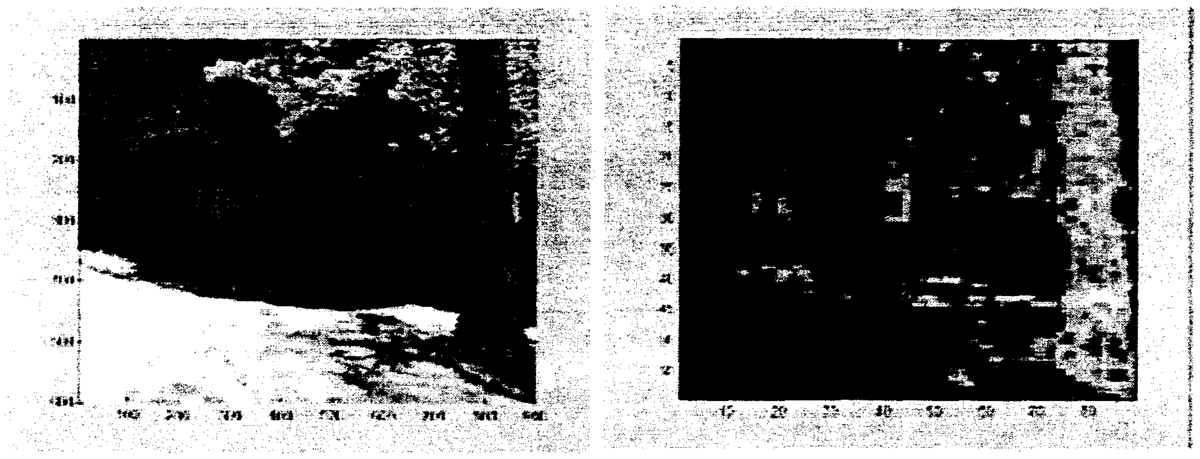


Figure 28. Results from automatically designed detectors for trees.

7.2 Reinforcement Learning Techniques for Segway

This research focused on machine learning, specifically testing reinforcement learning techniques on real-world robots. Reinforcement learning is a promising algorithm that has been tested in many simulated environments but on a limited basis in real-world scenarios. This research measures how well reinforcement-learning techniques perform when applied to real-world tasks, managed as a discrete-event dynamic system. This work will test reward anticipation models and reinforcement methods on a robot to determine if it can learn through reinforcement to complete a novel task.

The focus is on a specific navigation task using our Segway RMP. Reward signals will be used as reinforcement during the trials. In order to create a robot that can learn to associate reward as a consequence of its behavior the robot must be able to explore its environment and exploit its existing knowledge. These two objectives of exploration and exploitation are key factors in reinforcement learning [Sutton 1998]. Reinforcement learning (RL) methods including temporal difference learning will be used in the construction of the robot's learning algorithms [Rosenstein 2003].

The premise of the Segway RL experiment is to have the Segway learn how to travel from an initial position on the third floor of Featheringill Hall on the campus of Vanderbilt University to the third floor elevators, take an elevator to the ground floor, and travel from the ground floor through a set of doors to an outside patio. Tests can also be run to challenge the architecture's ability to re-plan when a path is completely blocked, such as when doors block off a portion of a hallway. RL methods take advantage of exploring the environment while choosing the actions predicted as most rewarding. Therefore, a robot using RL can learn the optimal path to a goal but also retains the means to explore different solutions. In reinforcement learning experiments the *states* are mapped to *actions* that connect between the next states that define a *policy* for the robot to use for a given task. In this experiment, states must be defined at positions along the task that break up the journey into segments connected by simple actions (i.e. basic behaviors). The possible variables of raw data that can be used to define the states include: distance readings from Segway's wheel odometers, distance readings from a SICK LMS laser rangefinder, and the angle of a particular distance reading from the laser scans. Preliminary data files have been recorded using the real Segway robot in a real-world environment that will be the basis for defining the states of the environment for the reinforcement learning trials on the Segway. Gazebo and Player/Stage software are also used as simulation tools to test experiments before running trials with the Segway in the real world. Stage has been used as an environment for other RL experiments with a simulated Segway RMP [Provost 2004], which encouraged the desire to go a step further in this experiment and use RL in the real world to test how well the theory holds in a real environment compared to a simulated one.

The framework of the experiment is made of the states, actions, rewards, and environment. In this experiment the states are defined both by odometry measurements and laser scans. The actions available to the Segway RMP at any given state are to move forward, backward, turn left, or turn right. The reward is based on the state and the time elapsed from the beginning of the trial. The states are derived from the environment; which includes the third floor of Featheringill Hall, a set of elevators on the third floor, the ground floor of the same building, and an outdoor patio next to the building. A large, positive reward (400) is sent if the state is a goal state. A significant, negative reward (-50) is sent if the state ends the experiment prematurely, without the goal being met. A small, negative reward is given for all other states based on the time that has elapsed in order to encourage learning a path that minimizes the time taken to reach the goal.

8. Summary

Our research achievements under the DARPA grant contributed significantly in the following areas:

Basic Research

- Multi-agent based robot control architecture called the Intelligent Machine Architecture (IMA)
    - The Vanderbilt team received a Space Act Award for this research from NASA JSC in October 2004.

- Cognitive Control and the Self Agent
    - Cognitive control in human is the ability to consciously manipulate thoughts and behaviors using attention to deal with conflicting goals and demands. We have been updating the IMA Self Agent towards this goal. If opportunity arises, we would like to work with NASA to empower Robonaut to do cognitive control.

Applications

- SES for Robonaut
    - Dr. Peters spent last two summers at JSC transferring our SES software to Robonaut.

- Robonaut Fault Diagnostic System
    - The Vanderbilt team in conjunction with Bill Bluethman and Ken Adler at JSC successfully developed a fault diagnostic system for Robonaut and tested on site.

- ISAC Behavior Generation and Learning
    - With an assistance from Drs. Matarić and Jenkins (see the USC report), the Vanderbilt team successfully implemented behavior generation and learning algorithm and tested on ISAC.

- Segway Research

    - We installed NDDS on the Segway and tested SES-based Segway navigation. The work is still in early development stage.

References

[Albus 1991]	J. S. Albus, Outline for a Theory of Intelligence, *IEEE Transactions on Systems, Man, and Cybernetics,* Vol. 21, No. 3, pp. 473–509, May 1991.
[Ambrose 2004]	R.O. Ambrose and C.G. Ambrose, Primate Anatomy, Kinematics and Principles for Humanoid Design, *International Journal of Humanoid Robotics*, Vol. 1, No. 1, pp. 175-198, World Scientific Publishing, March 2004.
[Ambrose 2000]	R. O. Ambrose, H. Aldridge, R. S. Askew, R. R. Burridge, W. Bluethmann, M. Diftler, C. Lovchik, D. Magruder, and F. Rehnmark, "Robonaut: Nasa's Space Humanoid," *IEEE Intelligent Systems,* Vol. 15, No. 4, pp. 57–63, July 2000.
[Carter 2000]	R. Carter, *Mapping the Mind*, University of California Press, 2000.
[Cohen 2001]	P.R. Cohen and N. Adams, An Algorithm for Segmenting Categorical Time Series into Meaningful Episodes, *Proc. Fourth Symp. on Intelligent Data Analysis,* Vol. 2189, pp. 198-207, 2001.
[Edmondson 1987]	C. Edmondson, *A Fuller Explanation: the synergetic geometry of R. Buckminster Fuller.* Boston: Birkhauser Verlag, Jan. 1987.
[Erol 2003]	D. Erol, J. Park, E. Turkay, K. Kawamura, O.C. Jenkins and M.J. Mataric, Motion Generation for Humanoid Robots with Automatically Derived Behaviors, *Proc. of IEEE Int'l. Conf. on Systems, Man, and Cybernetics,* Washington, DC, pp.1816-1821, Oct 2003.
[Franklin 2003]	S. Franklin, IDA: A Conscious Artifact?, Journal of Consciousness Studies, Vol. 10, No. 4-5, 2003.
[Hambuchen 2004]	K. A. Hambuchen, *Multi-Modal Attention and Event Binding in Humanoid Robots Using a Sensory EgoSphere*, Ph.D. Dissertation, Vanderbilt University, May 2004.
[Jenkins 2002]	O. C. Jenkins and Matarić, M. J., Deriving Action and Behavior Primitives from Human Motion Data, *Proc. 2002 IEEE/RSJ Int'l. Conf. on Intelligent Robots and Systems*, Lausanne, Switzerland, pp. 2551-2556, 2002.
[Kawamura 2004]	k. Kawamura, R.A. Peters II, R. Bodenheimer, N. Sarkar, J. Park, A. Spratley, and K. A. Hambuchen, Multiagent-based Cognitive Robot Architecture and its Realization, *International Journal of Humanoid Robotics,* Vol. 1, No. *1*, pp. 65-93, March 2004.
[Kawamura 2003]	K. Kawamura, D. C. Noelle, K. A. Hambuchen, T. E. Rogers and E. Turkay, A Multiagent Approach to Self-Reflection for Cognitive Robotics, *Intl. Conf. on Advanced Robotics*, Coimbra, Portugal, June 30-July 3, 2003, 568-575, 2003.
[Kawamura 2002]	K. Kawamura, A. B. Koku, D. M. Wilkes, R. A. Peters II, and A. Sekmen, Toward egocentric navigation, *International Journal of Robotics and Automation,* Vol. 17, No. 4, pp. 135–145, Oct. 2002.

[Kawamura 2000]	K. Kawamura, R. A. Peters II, D. M. Wilkes, W. A. Alford, and T. E. Rogers, ISAC: foundations of human-humanoid interaction, *IEEE Intelligent Systems*, Vol. 15, No. 4, pp. 38–45, July 2000.
[Kawamura 1986]	K. Kawamura, R.T. Pack, M. Bishay, and M. Iskarous, Design philosophy for service robots, *Robotics and Autonomous Systems*, *International Workshop on Biorobotics: Human-Robot Symbiosis*, (Eds. K. Kawamura and T.E. Davis), Elsevier, PP. 109-116, 1986.
[Koku 2003]	A.B. Koku, *Egocentric Navigation and its Applications*, Ph.D. Dissertation, Vanderbilt University, Nashville, TN, May 2003.
[Maes 1990]	P. Maes and R. A. Brooks, Learning to Coordinate Behaviors, *Proc. Eighth National Conf. on Artificial Intelligence*, Vol. AAAI-90, pp.796-802, 1990.
[Matarić 1992]	M. J. Matarić, Integration of Representation Into Goal-Driven Behavior-Based Robots, *IEEE Trans. Robotics and Automation*, Vol. 8, No. 3, pp. 304-312, June 1992.
[Miller 2003]	E.K. Miller, *Cognitive Control: Understanding the brain's executive*, in *Fundamentals of the Brain and Mind, Lecture 8*, June 11-13, 2003.
[Pack 1998]	R.T. Pack, *IMA: The Intelligent Machine Architecture*, PhD Dissertation, Vanderbilt University, Nashville, TN, May 1998.
[Peters 2003]	R.A. Peters II, K.A. Hambuchen, R.E. Bodenheimer, The Sensory EgoSphere: a mediating interface between sensors and cognition, In-house publication, December 2003.
[Peters 1999]	R. A. Peters II, D. M. Wilkes, D. M. Gaines, and K. Kawamura, "A software agent-based control system for human-robot interaction," in *Proceedings of 2nd Int'al Symp. Humanoid Robotics*, Tokyo, Oct. 1999.
[Pfeifer 1997]	Pfeifer, R. and C. Scheier, Sensory-motor coordination: the metaphor and beyond, *Robotics and Autonomous Systems*, Special Issue on "Practice and Future of Autonomous Agents," Vol. 20, No. 2-4, pp. 157-178, 1997.
[Provost 2004]	Provost J., Kuipers B. J., and Miikkulainen R. *Self-Organizing Perceptual and Temporal Abstraction for Robot Reinforcement Learning*. AAAI Workshop of Learning and Planning in Markov Processes, 2004.
[Rose 1998]	C. Rose, M. Cohen, and B. Bodenheimer, Verbs and Adverbs: Multidimensional motion interpolation, *IEEE Computer Graphics and Applications*, Vol. 18, No. 5, pp. 32-40, Sept. 1998.
[Rosenstein 2003]	M.T. Rosenstein and A.G. Barto. Supervised actor-critic reinforcement learning. In *Learning and Approximate Dynamic Programming: Scaling Up to the Real World*. Wiley & Sons, Inc., New York, 2003.
[Shepard 1968]	D. Shepard, A two-dimensional interpolation function for irregularly-spaced data, Proc. of the ACM National Conference, pp 517–524, ACM Press, 1968.

[Skubic 2004] M. Skubic, D. Noelle, M. Wilkes, K. Kawamura, and J. Keller, A Biologically Inspired Adaptive Working Memory for Robots, AAAI Fall Symposium, Workshop on the Intersection of Cognitive Science and Robotics: From Interfaces to Intelligence, Washington, DC, Oct 2004.

[Sutton 1998] Sutton, R. S. & Barto, A. G., *Reinforcement Learning. An Introduction.* Cambridge, MA: MIT Press, 1998.

[Urner 1991] K. Urner, The Invention Behind the Inventions: Synergetics in the 1990's, *The Synergetica Journal,* Vol. 1, No. 1, 1991.

[Wolpert 1998] D.M. Wolpert and M. Kawato, "Multiple Paired Forward and Inverse Models for Motor Control", *Neural Networks*, Vol. 11, pp.1317-1329, 1998.

[Yarger 1999] R. J. Yarger, G. Reese, and T. King, *MySQL and mSQL.* O'Reilly and Associates, August 1999.

Center for Intelligent Systems: Papers Published under MARS2020 Program

Park, J., S. Ferguson, N. Sarkar, K. Kawamura, R. Ambrose, B. Bluethmann, and K. Alder, Toward Intelligent System Health Monitoring for NASA Robonaut, IEEE-RAS/RSJ International Conference on Humanoid Robots (Humanoids 2004), Los Angeles, CA, November 10-12, 2004.

Kawamura, K., W. Dodd, and P. Ratanaswasd, Robotic Body-Mind Integration: Next Grand Challenge in Robotics, Invited Paper, *IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN)*, Kurashiki, Japan, September 20-24, 2004.

Park, J., P. Ratanaswasd, E.E. Brown Jr., T.E. Rogers, K. Kawamura, and D.M. Wilkes, Towards a Personal Robotic-aid System, *International Journal of Human-Friendly Welfare Robotic Systems*, Vol. 5, No. 2, pp. 2-12, June, 2004.

Peters II, R.A. and X. Ao, Acquisition of Topological Action Maps through Teleoperation, *Submitted to IEEE International Conference on Robotics and Automation (ICRA)*, April 26 - May 1, 2004, New Orleans, LA.

Kawamura, K., R.A. Peters II, R. Bodenheimer, N. Sarkar, J. Park, A. Spratley, and K. A. Hambuchen, Multiagent-based Cognitive Robot Architecture and its Realization, *International Journal of Humanoid Robotics, Vol. 1, No. 1*, pp. 65-93, March 2004.

Kawamura, K., T.E. Rogers, K. A. Hambuchen and D. Erol, Towards a Human-Robot Symbiotic System, *Robotics and Computer Integrated Manufacturing*, Vol. 19, pp. 555-565, 2003.

Erol, D., J. Park, E. Turkay, K. Kawamura, O.C. Jenkins and M.J. Mataric, Motion Generation for Humanoid Robots with Automatically Derived Behaviors, *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Washington, DC, October 2003, pp. 1816-1821, 2003.

Peng, J. and R.A. Peters II, Extract Salient Visual Features from Imagery-Motor Sequences for Mobile Robot Navigation, *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Washington, DC, October 5-8, 2003, pp. 2059-2064, 2003.

Kawamura, K., D.C. Noelle, K.A. Hambuchen and T.E. Rogers, "A Multi-Agent Approach to Self-Reflection for Cognitive Robots", *11th International Conference on Advanced Robotics (ICAR)*, Coimbra, Portugal, June 30 - July 3, 2003, pp. 568-575, 2003.

Kawamura, K. and D. Noelle, Requirements for Cognitive Robots, Invited Paper, *3rd International (IARP) Workshop*, Tsukuba, Japan, December 11-12, 2002.

Dissertations

Anthony Alford, Design and Implementation of the Self Agent System in a MultiAgent-Based Robot Control Architecture, Ph.D. Dissertation, Vanderbilt University, May 2003.

Kimberly A. Hambuchen, Multi-Modal Attention and Event Binding in Humanoid Robots Using a Sensory EgoSphere, Ph.D. Dissertation, Vanderbilt University, May 2004.

Carlotta A. Johnson, Enhancing a Human-Robot Interface Using a Sensory Egosphere, Ph.D. Dissertation, Vanderbilt University, May 2003.

Phongchai Nilas, A MultiAgent Based Architecture for an Adaptive Human-Robot Interface, Ph.D. Dissertation, Vanderbilt University, May 2003.

Jian Peng, Extraction of Salient Features from Sensory-Motor Sequences for Mobile Robot Navigation, Ph.D. Dissertation, Vanderbilt University, May 2004.

Tamara E. Rogers, The Human Agent: A Model for Human-Robot Interaction, Ph.D. Dissertation, Vanderbilt University, August 2003.

Master's Thesis

Xinyu Ao, Navigation via Sensory Motor Coordination, Master's Thesis, Vanderbilt University, December 2003.

Christina L. Campbell, Learning through Teleoperation on Robonaut, Master's Thesis, Vanderbilt University, December 2003.

Duygun Erol, Development of Procedural Memory for Humanoid Behavior and Task Learning, Master's Thesis, Vanderbilt University, August 2003.

Roberto E. Olivares, The Intelligent machine architecture version 2.5: a revised development environment and software architecture, Master's Thesis, Vanderbilt University, May 2003.

Juan Rojas, Sensory Integration with Articulated Motion on a Humanoid Robot, Master's Thesis, Vanderbilt University, May 2004.

Li Sun, A Binocular Vision System for a Humanoid Robot, Master's Thesis, Vanderbilt University, May 2004.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* 03-14-2005	2. REPORT TYPE **Final Report**	3. DATES COVERED *(From - To)* 08-01-2002 - 12-31-2004
4. TITLE AND SUBTITLE **Acquisition of Autonomous Behaviors by Robotic Assistants**		5a. CONTRACT NUMBER N/A
		5b. GRANT NUMBER NAG9-1446 NASA JSC
		5c. PROGRAM ELEMENT NUMBER N/A
6. AUTHOR(S) Kazuhiko Kawamura (PI) Alan Peters (Co-PI) Nilanjan Sarkar (Co-PI) Bobby Bodenheimer (Co-PI)		5d. PROJECT NUMBER N/A
		5e. TASK NUMBER N/A
		5f. WORK UNIT NUMBER N/A
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Vanderbilt University Division of Sponsored Research 110 21st Ave S, Suite 937 Nashville TN 37023		8. PERFORMING ORGANIZATION REPORT NUMBER N/A
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) NASA – Johnson Space Center Financial Management Office ATTM: LF231/Connie McDonald Houston TX 77058		10. SPONSOR/MONITOR'S ACRONYM(S) NASA-JSC
		11. SPONSOR/MONITOR'S REPORT NUMBER(S) N/A

12. DISTRIBUTION / AVAILABILITY STATEMENT
This report is unclassified and may be considered open for public knowledge.

13. SUPPLEMENTARY NOTES
N/A

14. ABSTRACT

Our research achievements under the NASA-JSC grant contributed significantly in the following areas.

Basic Research
Multi-agent based robot control architecture called the Intelligent Machine Architecture (IMA) : The Vanderbilt team received a Space Act Award for this research from NASA JSC in October 2004.
Cognitive Control and the Self Agent : Cognitive control in human is the ability to consciously manipulate thoughts and behaviors using attention to deal with conflicting goals and demands. We have been updating the IMA Self Agent towards this goal. If opportunity arises, we would like to work with NASA to empower Robonaut to do cognitive control.

Applications
1. SES for Robonaut, 2. Robonaut Fault Diagnostic System, 3. ISAC Behavior Generation and Learning, 4. Segway Research

15. SUBJECT TERMS
robotics, behavior-based robotics, robot control, artificial intelligence, machine learning, sensory-motor coordination, Robonaut

16. SECURITY CLASSIFICATION OF: U			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON **Kaz Kawamura / Flo Fottrell**
a. REPORT U	U	U	UU	45	19b. TELEPHONE NUMBER *(include area code)* **1.615.343.0697**