A Framework for Learning Declarative Structure

Stephen Hart, Shichao Ou, John Sweeney, and Rod Grupen
Department of Computer Science
University of Massachusetts
Amherst, MA 01003 USA
{shart, chao, sweeney, grupen}@cs.umass.edu

***Abstract*— This paper provides a framework with which a humanoid robot can efficiently learn complex behavior. In this framework, a robot is rewarded by learning how to generate novel sensorimotor feedback—a form of native motivation. This intrinsic drive biases the robot to learn increasingly complex knowledge about itself and its effect on the environment. The framework includes a mechanism for uncovering hidden state in a well-structured state and action space. We present an example wherein the robot, Dexter, learns a sequence of manual skills: (1) searching for and grasping an object, (2) the length of its arms, and (3) how to portray its intentions to human teachers in order to induce them to help.**

I. Introduction

Behavior can be decomposed into *declarative* and *procedural* components. The procedural structure captures knowledge about how to implement an abstract policy in a particular setting; e.g., use the left hand and an enveloping grasp [1]. The declarative structure captures abstract knowledge about the task; e.g., to pick up an object, we must first find the object, reach to it, and then grasp it. In this paper, we present a framework for a robot to autonomously learn the declarative structure.

The robot is motivated to explore new behavior in order to cause stimuli on multiple channels of sensorimotor feedback. This formulation provides the robot with native reward for transforming exterior phenomena into new patterns of interior stimulation. As a result, the robot accumulates knowledge regarding itself and the environment using a representation grounded in patterns of sensory and motor signals.

In many cases, the value of actions is obscured by hidden state in the sensorimotor feedback. We present an algorithm to resolve hidden state by finding sensorimotor variables that reduce entropy in the distribution of probable outcomes produced by action.

In this paper, we employ a platform for experimental research in bi-manual manipulation called "Dexter" to explore the development of behavior. Dexter has a 2-DOF pan/tilt head equipped with two Sony cameras capable of stereo triangulation and two 7-DOF whole-arm manipulators (Barrett Technologies, Cambridge MA). Each arm is equipped with a 3-finger, 4 DOF Barret Hand and 6-axis finger-tip load cells.

II. The Control Basis

The architecture presented in this paper employs the control basis framework for discrete event dynamic systems [2], [3]. This approach is designed to provide a combinatoric basis

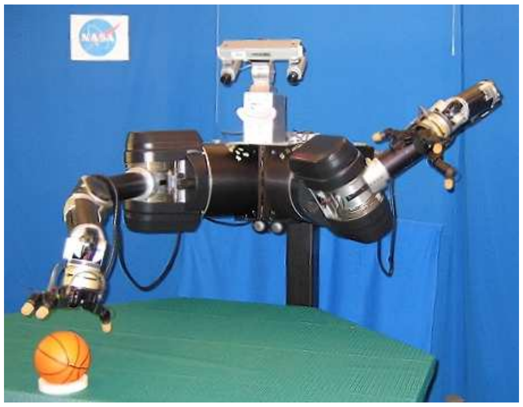
Fig. 1. Dexter, the UMass bi-manual humanoid.

for control that supports the representation of declarative and procedural knowledge. Primitive actions are closed-loop controllers that consist of an objective function $\phi \in \Omega_\phi$, a subset of available "sensor" resource abstractions $\sigma \in \Omega_\sigma$, and a subset of available "effector" resource abstractions $\tau \in \Omega_\tau$. A fully specified controller with its sensor and effector resources is denoted ϕ^σ_τ. The sequence of objective functions invoked captures the declarative structure of a task, while the σ and τ parameters capture procedural details appropriate for the run-time context.

In the control basis, concurrent control commands are constructed by projecting the output of a subordinate controller, ϕ_2, into the nullspace of a higher priority controller, ϕ_1[1]. The nullspace $\mathcal{N}_1$ of the control command of ϕ_1 is $(I - J_1^+ J_1)$ where J_i is the Jacobian matrix of the objective with respect to the configuration variables θ and J_i^+ is its pseudoinverse [4]. Our shorthand for this nullspace projection is written using the "subject-to" operator "$\triangleleft$" [2]. For example, $\phi_2 \triangleleft \phi_1$ captures the case where subordinate controller ϕ_2 projects outputs into the nullspace of the superior controller ϕ_1.

A. Discrete Event Dynamic Systems

The control basis also provides for a discrete state representation that reflects the status of underlying control tasks. Closed-loop controllers provide asymptotically stable behavior that is robust to local perturbations. The error dynamics

[1]Unless otherwise noted, we use ϕ_i to represent a particular instantiation of a controller with defined objective, sensor, and effector resources.

of a controller (e, $\dot{e}$) support a natural discrete abstraction of the continuous underlying state space. According to this discrete abstraction, *control events* are defined by recognizable temporal patterns in the control error.

The dynamic state of a controller ϕ_i can be characterized by a predicate p_i. In this paper we define three cases:

$$p_i \;=\; \begin{cases} -1 & : \;\; e_i \text{ is undefined} \\ 0 & : \;\; \dot{e}_i < \epsilon_\phi \\ 1 & : \;\; \dot{e}_i \geq \epsilon_\phi, \end{cases} \qquad (1)$$

where ϵ_ϕ is some small negative constant. A value of X for predicate p_i characterizes an aggregate "don't care" value over the three possible values and is often useful for state abstraction. For asymptotically stable controllers, for which $e(t)$ can be considered a Lyapunov function, $\dot{e}$ is negative definite. A "schema" designates a set of possible actions and a policy for selecting actions on the basis of patterns in the feedback that they reveal. Two such schema will be used in the experiments to follow: SEARCH-TRACK and GRASP.

SEARCH-TRACK is a schema constructed out of two primitive, closed-loop controllers. Position controller ϕ_0 moves a pan/tilt stereo head to a random configuration drawn from a context sensitive probability distribution. Another position controller, ϕ_1, moves a visual feature to the image center. Both of these objectives are addressed by actuating the pan/tilt motors of the stereo head. Together, they can be used to construct a variety of behavior. States $\mathcal{S} = (p_0, p_1)$ and actions ϕ_0 and ϕ_1 define a Markov Decision Process (MDP). A control policy can be defined over this MDP that finds and tracks a visual feature. Figure 2(a) shows the relevant states and non-zero state transitions for this schema. This schema results in a policy which moves the head to random pan/tilt locations by executing ϕ_0 until a feature is found. When this event occurs, the feature is tracked in the center of the image plane. The transition from state $(1, 0)$ to state $(1, -1)$ captures the situation in which the visual feature "outruns" the tracking controller.

Figure 2(b) shows a GRASP schema that employs actions ϕ_2 and ϕ_3 activated on states $\mathcal{S} = (p_2, p_3)$. Position controller ϕ_2 flexes the fingers of the robot hand toward a reference "closed" configuration. Force controller, ϕ_3, applies a reference force at each fingertip contact so as to generate a *wrench closure* condition that defines a grasp. Assuming that the hand begins in the "open" configuration, Figure 2(b) specifies that the hand close through the execution of ϕ_2. In the course of closing ϕ_3 will either assert state (1,-1) if there is no object to inhibit the movement, state (1,0) if contact with an object is made but the force condition is not met, or state (1,1) if an object is encountered and the force condition is achieved. Under the right conditions, the GRASP schema yields grasps that allow the hand to move the target object. Note that states $(1, -1)$ and $(1, 1)$ are *absorbing* in this policy because there are no transitions out of them.

B. Reward Structure

To guide the behavior of the robot, an *intrinsic* reward function is used to encourage multi-modal events in the robot's sensorimotor feedback. Equipped with this function, a robot will be biased toward behavior that causes the environment to change in discernible ways. In general, this function rewards a robot for bringing about control and sensory events that signify the status of useful objectives.

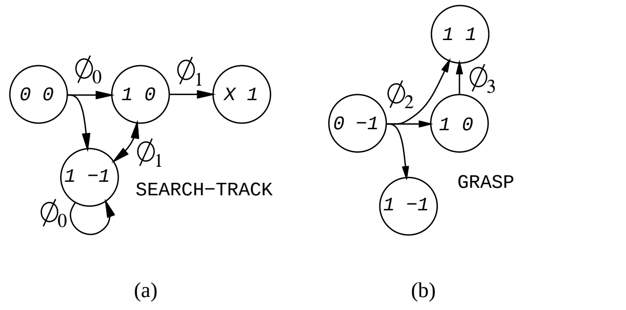

Fig. 2. Two simple sensorimotor schemas constructed from primitive closed loop controllers: (a) shows the SEARCH-TRACK schema that fi nds and tracks a visual feature using controllers ϕ_0 and ϕ_1; and (b) shows the GRASP schema that moves the fi ngers of a hand to a closed position using controllers ϕ_2 and ϕ_3. If an object is present, the fi ngers will envelop that object and take action ϕ_3 to grasp it.

For the experiments presented in this paper, a simple function is used pertaining to only a few possible sensory cues: a positive unit reward is given to the robot when new foreground regions appear in front of fixed backgrounds (a form of motion cue) and also when tactile events occur on the robot's fingertip sensors. A penalty of -0.2 was given for each action taken to decrease the number of steps taken in each trial.

C. Hierarchical Composition of Behavior

Behavioral schemas like those defined in Figures 2(a) and 2(b) can be reused hierarchically. We will use the notation Φ to represent a policy used as an action. For each behavior Φ_i, let $\mathcal{O}_i \subseteq \mathcal{S}_i$ be the set of absorbing states, and $\mathcal{A}_i$ be the action set of available controllers and schema. When in state s, the dynamic state of Φ_i is represented as a predicate p_i, according to the following three cases:

$$p_i \;=\; \begin{cases} -1 & : \;\; e_j \text{ is undefined } \forall \phi_j \in \mathcal{A}_i \\ 0 & : \;\; s \notin \mathcal{O}_i \\ 1 & : \;\; s \in \mathcal{O}_i. \end{cases} \qquad (2)$$

Adding hierarchy in this way allows for the transfer of behavior across tasks as well as an increased efficiency in learning. Coarticulation techniques for executing behavior schema concurrently with other schema or primitive controllers have been developed [5].

D. Learning Control Sequences

Reinforcement learning techniques such as Q-learning [6] can be employed to choose actions that allow an agent to cause rewarding transitions from one state to another [2], [7]. Q-learning estimates the action-value function $Q(s, a)$ for each state-action pair (s, a), where $s \in \mathcal{S}$ and $a \in \mathcal{A}$. Values are estimated by calculating the expected sum of discounted reward for each state-action pair. This update rule is defined as:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big(r(s,a) + \gamma \ \max_{a'} Q(s',a') - Q(s,a)\big)$$

where $\gamma \in [0,1]$ is the discount rate and $\alpha \in [0,1]$ is the learning rate. Reward function r is described in section II-B. With sufficient experience, this estimate is guaranteed to converge to the true value Q^*. A policy π maps states to actions. The *optimal* policy π^* maximizes the expected sum of future reward such that:

$$\pi^*(s) = \max_a Q^*(s,a)$$

E. Uncovering Hidden State in the MDP

In general, it is not possible to know all the state information required to make optimal, deterministic control decisions for a task *a priori*. However, in may cases hidden state is encoded in the observable sensorimotor features available to a robot. If we assume that the uncertainty in executing actions on the robot is small, then the main cause of stochasticity in the state transitions is due to factors that are not captured in the current state description. Therefore, an increase in the entropy of the state transitions indicates that the environment has changed in a way that is unmodeled in our current state representation. By attending to features that reduce entropy, and incorporating them into the state representation, the robot's actions become more deterministic. In this section, we provide a learning algorithm that monitors the entropy of the transition probability distribution and inspects the available feature space of sensorimotor data to uncover hidden state. When relevant information is found, a predicate corresponding to the dynamic state of a controller that tracks that information is added to the state representation.

Let $\mathcal{S} = (p_0, p_1, \ldots, p_n)$ be the factored state representation of control predicates. Let $\vec{F} \in \mathcal{F}$ be a feature vector of available sensorimotor data. Let the goal state set $\mathcal{G} \equiv \{s \in \mathcal{S}$ s.t. $r(s,a) > 0\}$. We can assign a label $l \in \{+, \circ\}$ to an episode according to whether it was rewarding or not as follows. If the agent reaches an absorbing state that is also a goal state $s \in \mathcal{G}$, we assign the label $l = +$. If the agent reaches an absorbing state $s \in \mathcal{O}$ that is not a goal state, we assign the label $l = \circ$. A general outline of the process used at each learning stage to uncover hidden state is given in Algorithm 1.

Algorithm 1 Uncovering Hidden State

```
1: Use Q-learning to find the optimal policy $\pi^*$ given the
   current state representation.
2: Use $\pi^*$ until it is determined that the entropy of a
   particular $s \times a$ transition distribution is high.
3: Using stochastic exploration with $\pi^*$, complete a number
   of trials and gather a data set $\mathcal{D}$, where each $d \in \mathcal{D}$ is a
   tuple $(\vec{F}, l)$.
4: Using $\mathcal{D}$, calculate a decision boundary, $g$, in feature
   space $\mathcal{F}$. Add a new controller $\phi_{n+1}$ that attends to the
   features of the decision boundary $g(\vec{F})$. Append the cor-
   responding predicate $p_{n+1}$ to $\mathcal{S}$ to form the new factored
   state representation.
5: Go to step 1.
```

III. Learning Trajectories

In this section, we present experimental results that show Dexter learning to search for, reach, and touch objects. At first, a human teacher presents an object to Dexter that is always within reach. In this setting, Dexter quickly learns a policy that will touch the object. Later, when the object is sometimes placed out of reach, Dexter learns to avoid penalties associated with actions by not reaching when the target is out of reach. Finally, Dexter learns that it can often "request" the human teacher to bring the object within its reach by executing certain actions with communicative content.

A. Dexter's Native Structure

In the following experiments, the object presented to Dexter is a small toy orange basketball, seen with Dexter in Figure 1, approximately 5 inches in diameter and made out of soft foam. A simple enveloping grasp reliably acquires the basketball.

During each trial, Dexter acquires a feature vector $\vec{F} \in \mathcal{F}$ of observable sensory signals. Background subtraction is used to segment foreground and background regions in Dexter's field of view. Foreground regions indicate motion relative to a fixed background. This kind of feature is determined to be attractive by design; it is a native, motion-based saliency cue that bootstraps Algorithm 1. $\mathcal{F}$ contains the first and second moments of foreground pixel distributions, the spatial location of the foreground objects relative to Dexter's coordinate frame, and the tactile response for each of Dexter's 6-axis finger-tip load cells.

We provided the robot with five native controllers: ϕ_0, ϕ_1, ϕ_2 and ϕ_3 from Section II-A—instantiated with the robot's pan/tilt head and hands—and position controller ϕ_4, which achieves a "Reach" behavior implemented with one of the robot's arms. We also provide the robot with the two schemas, SEARCH-TRACK and GRASP, presented in Figure 2.

We also provided Dexter with (prior learned) procedural details for allocating controller resources σ and τ [1]. When the object is placed on the robot's left side, the left arm is used to reach. When the object is placed on the robot's right side, the right arm is used to reach. It is the goal of the following experiments to learn declarative knowledge—how to choose actions that will reliably lead to the most accumulated reward.

B. Stage 1: Search-Reach-Grasp Behavior

The robot is provided with the action set $\mathcal{A} = \{\text{SEARCH-TRACK}, \text{GRASP}, \phi_4\}$. Providing the two schemas—rather than the corresponding primitive controllers—allows for encoding useful prior knowledge of the task. We will use a three-predicate state description $\mathcal{S} = (p_{st}, p_g, p_4)$, corresponding to the dynamic status of SEARCH-TRACK, GRASP, and ϕ_4. These three predicates provide a preliminary representation intended to guide the robot toward grasping behavior.

In the first experiment, consisting of 30 episodes, the robot learns a policy to acquire the basketball using Q-learning with an ϵ-greedy exploration strategy, where $\epsilon = 0.1$. As described in section II-B, a reward signal of 1 was given to the robot both when the object was found visually and when it was touched. A penalty of -0.2 was also given for each action taken. As seen in Figure 3, it takes the robot approximately ten trials to learn the optimal policy of $\pi^* = $ SEARCH-TRACK $\rightarrow$ "Reach" $\rightarrow$ GRASP, which corresponds to searching for the object, reaching to it, and grasping it. In this figure, the total accumulated reward for each episode is shown as well as the "task"-reward (what we know as the "goal" event of grasping the object, plus the penalty). Episodes 30-35 show a testing phase in which exploration was turned off.

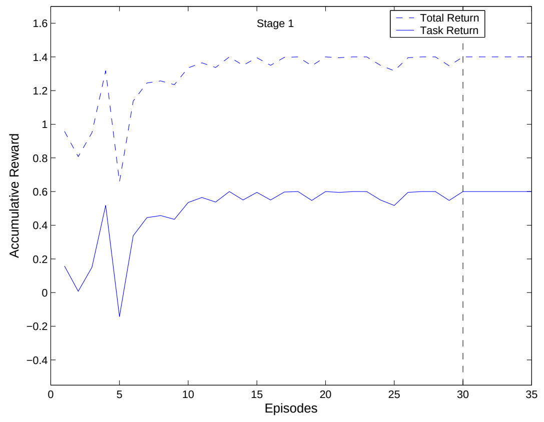

Fig. 3. Accumulated reward for Dexter learning search-reach-grasp policy averaged over 4 runs. The solid line shows the return for the touch reward and the penalty. The dotted line shows the total return for the search reward, the touch reward, and the penalty. Dexter learns π^* in about ten episodes.

C. *Stage 2: Learning Arm Length*

Once a policy was acquired for grasping the object, ten episodes (35-45) were performed in which, half the time, the object was placed out of Dexter's reach—a context that had never occurred before. Figure 4 shows how, using the policy from stage 1, the average reward drops to about half its former value. During episodes that Dexter could not reach the object, the negative penalty for taking actions still accumulated, but no final pay off for completing the task was rewarded. However, Dexter still gathers reward for finding the object visually.

Although position information is present in the sensory feedback, the robot is not attending to it. The current state representation, therefore, does not capture information pertaining to the "reachability" of the object. However, by computing a new decision boundary in $\mathcal{F}$, this state can be recovered and added to the state description; thus allowing the robot to learn not to reach if the object is unreachable. To uncover this information, the $\{+, \circ\}$ labels of episodes 35-45 were used to separate $\mathcal{F}$ using a support vector machine (SVM) [8]. Figure 5 shows the xy position of the object for these episodes, showing the clear cut at $x = 1.09$ meters. Given this information, a new position controller ϕ_5 can be constructed in which the position reference is the set of robot configurations where rewarding "touches" are forthcoming, i.e., $x < 1.09$m. With this addition, the robot is free to query the controller's state to establish the "reachable" states and modify its policy accordingly. The conjunction of the assertions (p_{st}, p_5) now resolves important hidden state about the task.

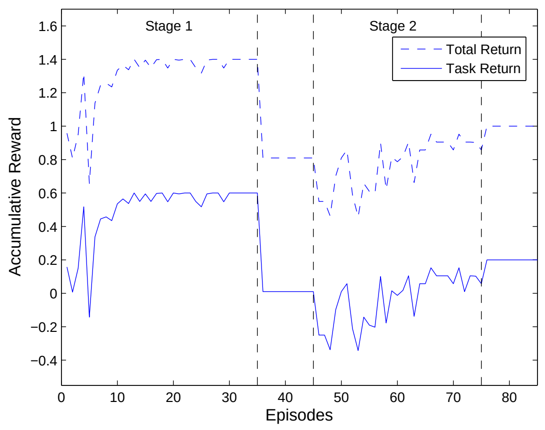

Fig. 4. Accumulated reward for the first two stages of learning. During stage 2, the object is placed outside of Dexter's workspace 50% of the time. When the predicate p_5 is added, the robot learns not to reach if the object is too far away. As a result, average reward increases.

Figure 4 shows how Dexter, after 30 more episodes (45-75), learns that if the object is too far away, not reaching is more rewarding on average. Episodes 75-85 show a testing phase in which exploration is turned off. If the object is reachable, Dexter will touch the object for a reward. But when the object is out of reach it has learned not to incur the penalty from a failed reach attempt. However, Dexter is unable, given the less constrained context, to achieve as much reward, on average, as was possible in stage 1.

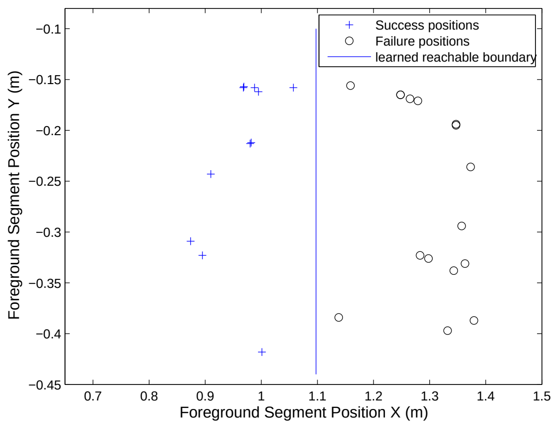

Fig. 5. The xy positions of the object for the trials seen in Figure 4, episodes 35-45. The decision boundary at $x = 1.09$m best separates the $\{+, \circ\}$ classes. This data set was generated by constraining the robot to use only its right arm. The learned boundary was transfered, through symmetry, to cases in which the procedural context dictated that Dexter use its left arm.

D. Stage 3: Learning When Humans Can Help

In the next stages of learning, seen in Figure 6, a human teacher appears in Dexter's field of view during each episode and the characteristics of the resulting foreground segment are added to the feature vector $\mathcal{F}$. In episodes 85-115, Dexter explores stochastic variations of the stage 2 policy and discovers a new context in which it can create a contingency for touching an object that is currently out of reach. When Dexter now reaches towards an out of reach object, the teacher moves the object closer. This action conveys the robot's intention to the human. This new context triggers a search in Algorithm 1, finding that the presence of a second, larger foreground object (as seen in Figure 7) often distinguishes the case where unreachable objects are ultimately graspable.

A new position controller ϕ_6 is configured to test the assertion that a large foreground object is present. In episodes 115-145, Dexter learns a new policy with the predicate p_6 added to the state description. Here, another unit of reward is received when Dexter finds a second foreground segment. This policy differs from the policy learned in stage 2 in that it causes Dexter to reach for unreachable objects in the presence of features associated with a nearby human, but not otherwise. For episodes 145-155, exploration is turned off to show the noticeable accumulation of more average reward then in stage 2. Note that, in stage 3, Dexter learns a policy that, on average, accumulates more reward than in stage 1. However, Dexter does not achieve the level of "goal" reward received under the constrained conditions of the first stage when the object was always placed within Dexter's reach.

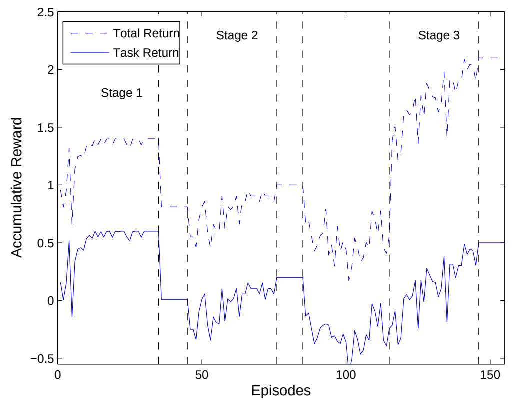

Fig. 6. Accumulated reward for all three stages of learning. When the predicate p_6 is added to the state representation, Dexter can exploit the presence of features associated with humans to achieve more reward.

IV. Discussion and Conclusions

We have presented a framework that provides a control theoretic basis for action, the hierarchical composition of behavior, an intrinsic drive towards skill acquisition, and a means for uncovering hidden state. Promising preliminary results have shown how Dexter achieves significant developmental milestones operating under this framework—including primitive

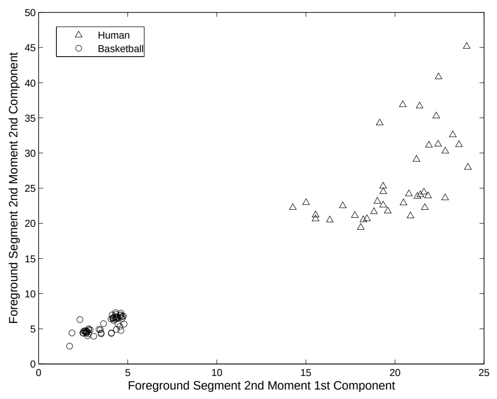

Fig. 7. The foreground objects' second moment (covariance) when the human is present. These values represent the first two principle components of pixel distributions on the left image plane. The distinct clusters correspond to the basketball and the nearby human.

communicative actions—but many questions pertaining to the full expressive power of the methodology remain unanswered.

It is possible that Algorithm 1 mayb introduce bias into the system as the teacher "directs" the robot's attention to particular sensorimotor features. However, rather than being a hindrance, we believe that this approach allows the teacher to convey meaningful information to the robot, while still allowing the robot to discern exactly how this information is related to its sensorimotor signals.

For future work, we wish to examine the tractability of learning declarative and procedural structure *simultaneously*. We also wish to explore, in more detail, both the autonomous acquisition of hierarchical behavior and the formulations multi-modal reward signals. A study of convergence properties of the overall system remains to be made.

Acknowledgments

This research was supported under contract numbers ARO W911NF-05-1-0396 and NASA NNJ05HB61A-5710001842 and under NASA Graduate Student Research Program fellowship NNJ05JG73H.

References

[1] S. Hart, R. Grupen, and D. Jensen, "A relational representation for procedural task knowledge," in *Proceedings of the American Association for Artificial Intelligence Conference*, 2005.

[2] M. Huber, "A hybrid architecture for adaptive robot control," Ph.D. dissertation, Department of Computer Science, University of Massachusetts Amherst, 2000.

[3] J. A. Coelho, "Multifingered grasping: Grasp reflexes and control context," Ph.D. dissertation, Department of Computer Science, University of Massachusetts Amherst, 2001.

[4] Y. Nakamura, *Advanced Robotics: Redundancy and Optimization*. Addison-Wesley, 1991.

[5] K. Rohanimanesh and S. Mahadevan, "Coarticulation: An approach for generating concurrent plans in markov decision processes," in *Proceedings of the 22nd International Conference on Machine Learning (ICML-2005)*, Bonn, Germany, 2005.

[6] D. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3,4, pp. 279–292, 1992.

[7] R. Sutton and A. Barto, *Reinforcement Learning*. Cambridge, Massachusetts: MIT Press, 1998.

[8] T. Joachims, "Learning to classify text using support vector machines," Ph.D. dissertation, Department of Computer Science, Cornell University, 2002.