Improving Grasp Skills Using Schema Structured Learning

Robert Platt
Dexterous Robotics Laboratory
Johnson Space Center
NASA
robert.platt-1@nasa.gov

Roderic A. Grupen
Laboratory for Perceptual Robotics
Department of Computer Science
University of Massachusetts, Amherst
grupen@cs.umass.edu

Andrew H. Fagg
Symbiotic Computing Lab
School of Computer Science
University of Oklahoma
fagg@ou.edu

***Abstract*— In the control-based approach to robotics, complex behavior is created by sequencing and combining control primitives. While it is desirable for the robot to autonomously learn the correct control sequence, searching through the large number of potential solutions can be time consuming. This paper constrains this search to variations of a generalized solution encoded in a framework known as an *action schema*. A new algorithm, SCHEMA STRUCTURED LEARNING, is proposed that repeatedly executes variations of the generalized solution in search of instantiations that satisfy action schema objectives. This approach is tested in a grasping task where Dexter, the UMass humanoid robot, learns which reaching and grasping controllers maximize the probability of grasp success.**

I. INTRODUCTION

In contrast to the sense-think-act paradigm, control-based and behavior-based approaches to robotics realize desired behavior by sequencing and combining primitive controllers or behaviors. These approaches depend heavily on a higher-level decision-making mechanism that selects the correct sequence or combination of primitives to execute. One way to automatically learn the correct sequence of primitives is to encode the problem as a Markov Decision Process, and solve it using Reinforcement Learning [1], [2]. However, in the absence of an *a priori* model of controller performance, this approach requires the robot to explore the effects executing every action in every state. Although the system designer can constrain the potential action choices [1], the need to explore a large number of actions can slow down learning. This paper addresses this problem by encoding a generalized solution as an *action schema*. Learning speed is increased by constraining the system only to consider variations of this generalized solution. A new algorithm called SCHEMA STRUCTURED LEARNING discovers which instantiations of the action schema are appropriate in different problem contexts. This paper explores this approach in the context or reaching and grasping.

Our approach is based on the notion of a *schema*, first proposed in the psychology literature by Piaget. Piaget loosely defines a schema to be a mental representation of an action or perception [3]. Through the process of *assimilation*, the child adapts an existing schema to incorporate new experiences, encoding these experiences as variations on the same general structure. In what Piaget calls a circular reaction, the child is motivated to reproduce behavior that results in activation of a reflex or an "interesting" result. Piaget's model is an inspiration for the current paper. In a process similar to the circular reaction, this paper proposes that the robot repeatedly executes variations of the general behavior encoded by the action schema. By observing the effects of different instantiations of the action schema, the robot learns how to produce the desired behavior in different problem contexts.

Arbib's *schema theory* brings the notion of the schema into a computational framework. At the lowest level, schema theory proposes two major types of schemas: the perceptual schema and the motor schema [4]. A perceptual schema is a process that responds to only a specific, task-relevant concept. A motor schema is a generalized program that describes how to accomplish a task. When a perceptual schema triggers that its target concept is present, it can "give access" to a motor schema that takes the appropriate action. Schema theory proposes that a large number of perceptual schemas and motor schemas can interact in the context of a *coordinated control program*, thereby generating intelligent behavior [4]. Arkin applies some of these ideas to behavior-based robotics [2].

Gary Drescher also develops a schema-based approach to intelligent behavior. Drescher's *schema mechanism* appropriates the Piagetian schema as its fundamental building block and develops a complex framework for computationally learning new concepts, *items*, and hypothesizing new schemas to interact with these concepts [5]. Learning starts with a few schemas and *primitive* items that represent basic motor activities and perceptions. By executing schemas, the system discovers new items and proposes new schemas for interacting with these items.

This paper proposes a new approach to robot learning based on a generalized representation of robot behavior known as the *action schema*. The action schema may be *instantiated* by specific implementations of the generalized behavior. An instantiation is considered to *succeed* or *fail* depending upon whether it results in desired state transitions specified by the action schema. A new on-line learning algorithm, SCHEMA STRUCTURED LEARNING, is proposed that explores different instantiations and discovers which instantiations are most likely to succeed through a process of trial-and-error. This paper explores the action schema approach in the context of

robotic reaching and grasping. Generalized grasping behavior is represented by a LOCALIZE-REACH-GRASP action schema. SCHEMA STRUCTURED LEARNING discovers how to select appropriate reach and grasp control actions based on coarse visual information including object location, orientation, eccentricity, and length. A series of experiments are reported where Dexter, the UMass bimanual humanoid robot, attempts to grasp an object using various different reach and grasp primitives. The robot learns to select reach and grasp primitives that optimize the probability of a successful grasp. This paper is an expansion of our earlier work reported in [6]. The current paper better defines optimality in the context of the action schema and proposes a sample-based version of the algorithm.

In addition to describing controllers used for localizing, reaching, and grasping, Section II gives a brief overview of the *control basis* approach to robot behavior used in this paper. Sections III and IV describe the action schema framework, define the notion of the optimal policy instantiation, and give an algorithm, *schema structured learning*, for autonomously discovering these optimal instantiations. Finally, Section V presents experimental results that demonstrate *schema structured learning* to be a practical way of autonomously learning reaching and grasping behavior.

II. CONTROL-BASED REACHING AND GRASPING

When using a control-based approach, a framework is needed that allows controllers to be sequenced in an organized way. The *control basis* framework accomplishes this by organizing the set of viable controllers and providing a robust way of evaluating system state [1]. This section describes the control basis framework and details controllers that are used for localizing, reaching, and grasping.

The control basis can systematically specify an arbitrary closed-loop controller by matching an *artificial potential function* with a *sensor transform* and *effector transform* [1]. The potential function specifies controller objectives, the effector transform specifies what degrees of freedom the controller uses, and the sensor transform implements the controller feedback loop. For example, consider a REACH controller. The sensor transform specifies the goal configuration of the end-effector. The effector transform specifies what degrees of freedom are used to accomplish the task.

In general, the control basis realizes a complete controller by selecting one artificial potential from a set $\Phi = \{\phi_1, \phi_2, \ldots\}$, one sensor transform from a set $\Sigma = \{\sigma_1, \sigma_2, \ldots\}$, and one effector transform from a set $\Upsilon = \{\tau_1, \tau_2, \ldots\}$. Given Φ, Σ, and Υ, the set of controllers that may be generated is $\Pi \subseteq \Phi \times \Sigma \times \Upsilon$. When specifying a fully-instantiated controller, the notation $\phi_i|_\tau^\sigma$ denotes the controller constructed by parameterizing potential function ϕ_i with sensor transform σ and effector transform τ.

The control basis framework also allows composite controllers to be constructed that execute multiple constituent controllers concurrently. Each constituent controller is assigned a priority, and controllers with lower priority are executed in the nullspace of controllers with higher priority. Composite controllers are denoted, $\phi_b|_\tau^\sigma \triangleleft \phi_a|_\tau^\sigma$, where $\phi_b|_\tau^\sigma$ is said to execute "subject-to" (i.e. in the nullspace of) $\phi_a|_\tau^\sigma$.

System state is measured in terms of controller dynamics. At any point in time, the instantaneous error and the instantaneous gradient of error can be evaluated. Although the more general system dynamics can be treated [7], in this paper we will consider only controller convergence to establish system state. For example, the state of having grasped an object with some effector is represented by the convergence status of a grasp controller parameterized by that effector transform.

This paper creates grasping behavior by combining and sequencing controllers based on the LOCALIZE, REACH, and GRASP artificial potentials. One LOCALIZE controller is used in the experiments described in this paper. This controller, $\phi_l|_{\tau_{cam}}^{\sigma_l}$, segments the object and characterizes the resulting blob in terms of its three-dimensional Cartesian position, orientation, length, and eccentricity.

The REACH controllers that are used in this paper are referenced with respect to the last object position recovered by a LOCALIZE controller. There are two types of REACH controllers: $\phi_{rp}|_{\tau_y}^{\sigma_y(x)}$ and $\phi_{ro}|_{\tau_y}^{\sigma_y(\theta)}$. $\phi_{rp}|_{\tau_y}^{\sigma_y(x)}$ moves the centroid of the y contact set to a position offset of x from the object centroid along the object major axis. $\phi_{ro}|_{\tau_y}^{\sigma_y(\theta)}$ associates each contact with a line from the contact frame centroid through the contact itself. $\phi_{ro}|_{\tau_y}^{\sigma_y(\theta)}$ orients the manipulator so that the average angle between each contact's line and the object major axis (for the y set of contacts) is θ. If $\phi_{rp}|_{\tau_y}^{\sigma_y(x)}$ executes alone, then the robot reaches to the position while leaving orientation unspecified. If orientation control executes subject to position control, $\phi_{ro}|_{\tau_y}^{\sigma_y(\theta)} \triangleleft \phi_{rp}|_{\tau_y}^{\sigma_y(x)}$, then the robot reaches to the desired position while also attempting to achieve the desired orientation.

GRASP controllers displace contacts toward good grasp configurations using feedback control [8], [9]. This approach uses tactile feedback to calculate an error gradient and displace grasp contacts on the object surface without a geometric object model. After making light contact with the object using sensitive tactile load cells, the controller displaces contacts toward minima in the grasp error function using discrete probes [8] or a continuous sliding motion [10]. This paper uses two GRASP controllers: $\phi_g|_{\tau_{123}}^{\sigma_{123}}$ and $\phi_g|_{\tau_{12}}^{\sigma_{12}}$. $\phi_g|_{\tau_{123}}^{\sigma_{123}}$ uses three physical contacts to synthesize a grasp, while $\phi_g|_{\tau_{12}}^{\sigma_{12}}$ combines two physical contacts (out of three) into a *virtual finger* [11] that is considered to apply a single force that opposes a third physical contact.

III. THE ACTION SCHEMA

The action schema encodes generalized robot behavior that can be instantiated by a specified set of control policies. This is advantageous in the context of learning because it constrains the number of potential solutions that need to be considered. The generalized behavior, encoded by the action schema, is represented by a policy defined over an abstract state and action space. A one-to-many mapping is defined between the abstract state and action space and an *underlying* state and action space. This underlying space is assumed to represent

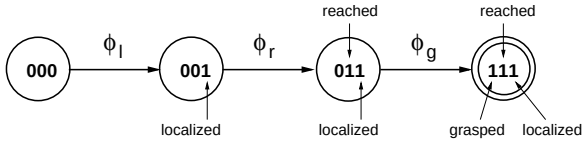


Fig. 1. The localize-reach-grasp action schema. The circles with binary numbers in them represent abstract states. The arrows represent abstract actions and possible transitions.

system state and action with the finest granularity available in the control representation. The mapping represents a set of underlying states by a single abstract state and corresponds each abstract action with a number of potential underlying action alternatives. This allows the action schema's abstract policy to be translated into a number of *policy instantiations* that define a set of potential solutions. The action schema also specifies an abstract transition function that defines desired transition behavior in the underlying space. The objective of schema structured learning is to discover which instantiations of an action schema's abstract policy are most likely to result in transitions that are consistent with the abstract transition function.

Let $S' \times A'$ be the abstract state-action space defined by the action schema, and let $S \times A$ be the underlying state-action space that represents possible robot behaviors. The abstract policy is a mapping from abstract states to abstract actions,

$$\pi' : S' \rightarrow A'. \tag{1}$$

This function deterministically specifies which abstract action the system should take in any given abstract state. For example, Figure 1 illustrates the LOCALIZE-REACH-GRASP action schema. The circles illustrate four abstract states that represent the generalized stages of the behavior. There are three abstract actions: ϕ_l (LOCALIZE), ϕ_{rp} (REACH), and ϕ_g (GRASP). The abstract policy, π', defines which abstract actions are to be taken from abstract states:

$$\begin{aligned} \pi'(000) &= \phi_l \\ \pi'(001) &= \phi_{rp} \\ \pi'(011) &= \phi_g. \end{aligned} \tag{2}$$

For example, if the robot is in abstract state (000), then π' executes abstract action ϕ_l.

The abstract policy is mapped to policy instantiations that implement the same qualitative type of behavior. This policy mapping is derived from state and action mappings that group together similar states and actions as follows. Let $f : S \rightarrow S'$ and $g : A \rightarrow A'$ be state and action mappings that uniquely assign each underlying state and action to an abstract state and action. The inverses of these functions are defined to be $f^{-1}(s') = \{s \in S | f(s) = s'\}$ and $g^{-1}(a') = \{a \in A | g(a) = a'\}$. g^{-1} maps each abstract action to an equivalence class of actions (i.e., controllers) that perform the same function in different ways. In the control basis representation, these controllers share the same artificial potential, but have different sensor and effector parameterizations. Similarly, f^{-1} maps each abstract state to an equivalence class of states. In the case of the LOCALIZE-REACH-GRASP action schema illustrated in Figure 1, the three abstract actions map to the various LOCALIZE, REACH, and GRASP controllers described in Section II:

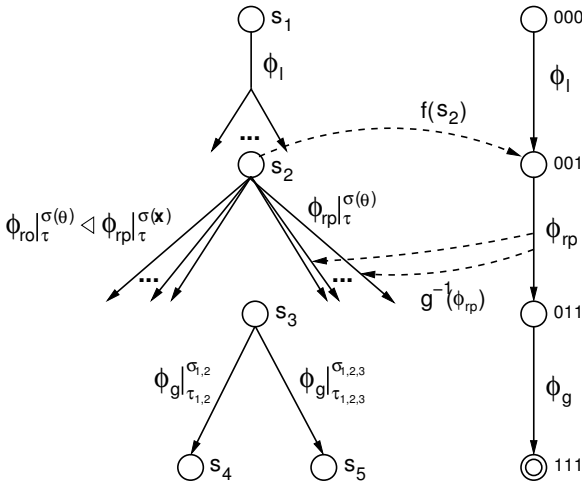


Fig. 2. Possible instantiations of the LOCALIZE-REACH-GRASP action schema. The inverse action mapping, g^{-1}, projects the abstract REACH action onto the set of possible REACH actions.

$$g^{-1}(\phi_l) = \{\phi_l|^{\sigma_l}_{\tau_{cam}}\}, \tag{3}$$

$$\begin{aligned} g^{-1}(\phi_r) = \{&\phi_{rp}|^{\sigma_{123}(x)}_{\tau_{123}}, \phi_{rp}|^{\sigma_{12}(x)}_{\tau_{12}}, \\ &\phi_{ro}|^{\sigma_{123}(\theta)}_{\tau_{123}} \triangleleft \phi_{rp}|^{\sigma_{123}(x)}_{\tau_{123}}, \\ &\phi_{ro}|^{\sigma_{12}(\theta)}_{\tau_{12}} \triangleleft \phi_{rp}|^{\sigma_{12}(x)}_{\tau_{12}}\}, \end{aligned} \tag{4}$$

$$g^{-1}(\phi_g) = \{\phi_g|^{\sigma_{123}}_{\tau_{123}}, \phi_g|^{\sigma_{12}}_{\tau_{12}}\}. \tag{5}$$

The state and action mappings above allow the abstract policy to be translated into a set of potential policy instantiations. This can be accomplished on a step-by-step basis by determining the set of actions that are consistent with the abstract policy in a given state. Assume that the system is in abstract state, s'_t. The abstract action specified by $\pi'(s'_t)$ can be projected onto a set of equivalent underlying actions using the inverse action mapping, $g^{-1}(\pi'(s'_t))$. Therefore, given the state and action mapping, the abstract policy, π', can be mapped onto any policy, π, such that,

$$\forall s_t \in S, \;\; \pi(s_t) \in B(s_t), \tag{6}$$

where

$$B(s_t) = g^{-1}(\pi'(f(s_t))). \tag{7}$$

These policies are called *policy instantiations* of the abstract policy. This is illustrated in Figure 2. Suppose that the robot is in state $s_2 \in S$. The state mapping, $f(s_2) = (001)$, projects this state onto $(001) \in S'$. From this abstract state, the abstract policy takes abstract action ϕ_{rp}, $\pi'(001) = \phi_{rp}$. Finally, the inverse action mapping, g^{-1}, projects

this abstract action onto the reach choices, $g^{-1}(\phi_{rp}) = \{\phi_{rp}|_{\tau_{123}}^{\sigma_{123}(x)}, \phi_{rp}|_{\tau_{12}}^{\sigma_{12}(x)}, \phi_{ro}|_{\tau_{123}}^{\sigma_{123}(\theta)} \lhd \phi_{rp}|_{\tau_{123}}^{\sigma_{123}(x)}, \phi_{ro}|_{\tau_{12}}^{\sigma_{12}(\theta)} \lhd \phi_{rp}|_{\tau_{12}}^{\sigma_{12}(x)}\}$. Although this enumeration of potential REACH controllers does not separately list controllers with different position or orientation offsets, ϕ_{rp} implicitly maps to a real-valued space of REACH controllers with different offsets.

The goal of schema structured learning is to discover the policy instantiation(s) that maximizes the probability of meeting transition constraints specified by the action schema. The action schema deterministically characterizes the desired behavior of the robot with the abstract transition function,

$$T' : S' \times A' \rightarrow S'. \tag{8}$$

This specifies how the system must transition in response to executing the action. When executing action $a \in A$ from state $s_t \in S$, the next state, $s_{t+1} \in S$, is consistent with the abstract transition function when the following holds true:

$$s_{t+1} \in N(s_t, a), \tag{9}$$

where

$$N(s_t, a) = f^{-1}(T'(f(s_t), g(a))). \tag{10}$$

In this expression, $f(s_t)$ and $g(a)$ translate the underlying state and action into their abstract equivalents. The abstract transition function, T', maps this abstract action into an expected next abstract state. Finally, f^{-1} translates the expected next abstract state into a set of underlying next states. As long as action $a \in A$ causes the robot to transition to one of these next states, the action is said to *succeed*. If the action causes a different transition, then the action *fails*. For example, consider the LOCALIZE-REACH-GRASP action schema illustrated in Figure 2 again. The abstract transition function,

$$\begin{aligned} T'(000, \phi_l) &= 001 \\ T'(001, \phi_{rp}) &= 011 \\ T'(011, \phi_g) &= 111, \end{aligned} \tag{11}$$

is encoded in the arrows in the action schema on the right. If the robot takes abstract action ϕ_{rp} from abstract state (001), the abstract transition function requires the system to transition to state (011). Suppose that the robot is in state $s_2 \in S$, and executes $\phi_{rp}|_{\tau_{12}}^{\sigma(x)_{12}}$. If the system does not transition to a state, $s_{t+1} \in S$, that maps to $(011) \in S'$, $f(s_{t+1}) = (011)$, then $\phi_{rp}|_{\tau}^{\sigma(x)}$ fails.

In schema structured learning, the robot continues to execute actions in accordance with the abstract policy until an action fails or an absorbing state is reached. In the LOCALIZE-REACH-GRASP action schema (Figure 1), (111) is an absorbing state.

Bringing these pieces together, an action schema is a represented as a tuple,

$$\mathcal{S} = \langle S', A', \pi', T' \rangle, \tag{12}$$

where S' is the abstract state set, A' is the abstract action set, π' defines the abstract policy, and T' defines the abstract transition function. When defining an action schema, we will require that the path implicitly specified by π' and T' does not contain cycles.

IV. Optimal Policy Instantiations

The abstract policy encoded by the action schema maps to many different policy instantiations. The goal of the system is to discover which policy instantiations maximize the probability of reaching the goal, while satisfying the action schema's transition contraints. This is called the *optimal* policy instantiation. This section defines the optimal policy instantiation and introduces the SCHEMA STRUCTURED LEARNING algorithm given in Table I.

A. Definition of the Optimal Policy Instantiation

The abstract transition function defines how the robot system must transition after executing actions. An action is said to *succeed* when it causes a transition that is consistent with the action schema abstract transition function. By induction, a state-action trajectory succeeds when each component action succeeds. An *optimal* policy instantiation, π^*, is one which maximizes the probability of a successful trajectory. Let $P^{\pi}(a|s_t)$ be the probability of a successful trajectory, given that the system takes action $a \in A$, starting in state $s_t \in S$, and follows policy instantiation π after that. If Π is defined to be the set of all possible policies, then

$$P^*(a|s_t) = \max_{\pi \in \Pi} P^{\pi}(a|s_t) \tag{13}$$

is the maximum probability of a successful trajectory taken over all possible policies. This allows the optimal policy to be calculated using,

$$\pi^*(s_t) = \arg \max_{a \in B(s_t)} P^*(a|s_t), \tag{14}$$

where $B(s_t) = g^{-1}(\pi'(f(s_t)))$ (Equation 7) is the set of actions that are consistent with the abstract transition function when the system is in state $s_t \in S$. The optimal policy always selects the action that maximizes the probability of satisfying action schema transition constraints.

B. Calculating an Optimal Policy Instantiation

Unfortunately, it is impractical to use Equations 13 and 14 directly to solve for the optimal policy instantiation, because the set of possible policy instantiations, Π, is exponential in the number of actions. However, as is the case with Markov Decision Processes (MDPs), this problem admits a dynamic programming algorithm that is polynomial in the number of viable state-action pairs. Whereas the dynamic programming approach to MDPs requires successive approximation of the true value function, it is possible to solve for the optimal policy instantiation of an action schema in a single "pass."

Recall that $P^*(a|s_t)$ is the maximum probability of a successful trajectory, given that the robot takes action $a \in A$ from state $s_t \in S$ and follows the optimal policy instantiation thereafter. This can be calculated recursively,

$$\begin{aligned} P^*(a|s_t) &= P(\text{success}|s_t, a) \\ &\sum_{s_{t+1} \in N(s_{t+1}, a)} T(s_{t+1}|s_t, a) \max_{a \in B(s_{t+1})} P^*(a|s_{t+1}), \end{aligned} \tag{15}$$

TABLE I

SCHEMA STRUCTURED LEARNING ALGORITHM

Function SCHEMA STRUCTURED LEARNING
1. Repeat
2. Get current state $s_t \in S$
3. Let $B(s_t) = D_{f(s_t)}(\pi'(f(s_t)))$
4. Evaluate $\pi^*(s_t) = \arg\max_{a \in B(s_t)} P^*(a|s_t)$
5. Execute $\pi^*(s_t)$
6. Get next state $s_{t+1} \in S$
7. Update transition model $P(\text{success}|s_t, a)$
8. Update sample set in $D_{f(s_t)}$ based on P^*
9. If action $\pi^*(s_t)$ failed, break from loop.
10. While $f(s_t)$ is not in an absorbing state.

where $P(\text{success}|s_t, a)$ is the probability that action a succeeds from state s_t, $N(s_{t+1}, a)$ is defined in Equation 10, and $T(s_{t+1}|s_t, a)$ is the probability that taking action a from state s_t causes the robot to transition to state s_{t+1} (notice the similarities to the standard Bellman equation.) If it is possible to deterministically characterize how *successful* actions transition, then this equation can be simplified,

$$P^*(a|s_t) = P(\text{success}|s_t, a) \max_{a \in B(T_s(s_t,a))} P^*(a|T_s(s_t, a)), \quad (16)$$

where

$$T_s : S \times A \rightarrow S, \;\; T_s(s_t, a) = s_{t+1} \quad (17)$$

is a function that deterministically characterizes how an action transitions if the action is assumed to be successful. Note that this does not mean that the underlying transition function must be deterministic in general - the outcome is deterministic only if the action succeeds.

C. Schema Structured Learning Algorithm

This approach to calculating the optimal policy instantiation is the basis of the SCHEMA STRUCTURED LEARNING algorithm given in Table I. Given an action schema and the appropriate mapping, this algorithm learns the optimal policy instantiation online through a trial-and-error process. Whereas many trial-and-error learning algorithms require substantial training, the structure imposed by the action schema framework makes SCHEMA STRUCTURED LEARNING practical for many real-life robot applications. As it is written in Table I, this algorithm executes one instantiation of the generalized behavior. Note that the algorithm must be executed multiple times in order to accumulate experience and learn.

A key feature of the version of SCHEMA STRUCTURED LEARNING given in Table I is that it is sample-based. Assume that the robot is in state, $s_t \in S$, and that $a' = \pi'(f(s_t))$. Instead of evaluating every instantiation of the abstract policy, this algorithm only evaluates a fixed number of samples from the set,

$$D_{f(s_t)}(a') \subseteq g^{-1}(a'). \quad (18)$$

This set is drawn from the probability distribution $P^*(a|s)$ marginalized over all states $s \in f^{-1}(s_t)$, thereby oversampling regions of the action space that are estimated to have a high probability of leading to a successful trajectory. Step 8 of SCHEMA STRUCTURED LEARNING resamples this set based on the latest approximation of P^*. This approach enables the algorithm to evaluate a real-valued action space (such as the REACH controller with position and orientation offsets) when selecting a policy instantiation to execute.

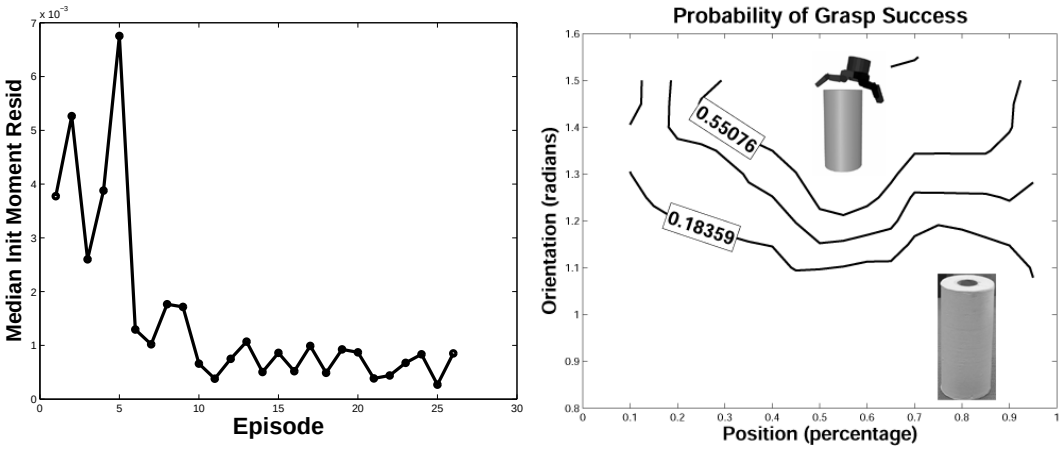
SCHEMA STRUCTURED LEARNING gains experience by repeatedly executing policy instantiations of the action schema. While the system initially executes random instantiations of the abstract policy, performance quickly improves. Assume that the robot is in state $s_t \in S$. In steps 3 and 4, the algorithm evaluates the probability of a successful trajectory, $P^*(a|s_t)$, using Equation 17 for all actions in $D_{f(s_t)}(\pi'(f(s_t)))$. By Equation 14, the action that maximizes this probability is part of an optimal policy instantiation (assuming that the sample set contains this action). Step 5 executes this action, and step 6 evaluates the outcome of the action. Step 7 incorporates this experience into the transition model. Step 8 re-samples $D_{f(s_t)}$ from the new estimate of P^*. Steps 9 and 10 continue to iterate taking actions until an action fails or an absorbing state is reached. SCHEMA STRUCTURED LEARNING must be repeatedly executed until learning is complete.

V. EXPERIMENTS

A series of eight experiments were performed that characterize the learning performance of SCHEMA STRUCTURED LEARNING in the context of grasp synthesis. In each experiment, Dexter attempted to localize, reach, and grasp a towel roll measuring 20cm high and 10cm in diameter 26 times. In these experiments, only three-fingered grasps, $\phi_g|^{\sigma_{123}}_{\tau_{123}}$, were allowed. At the beginning of each trial, the towel roll was placed vertically in approximately the same tabletop location. Then, SCHEMA LEARNING was executed using the LOCALIZE-REACH-GRASP action schema until either the absorbing state was reached or an action failed. In either case, the trial was terminated, the system was reset, and a new trial was begun.

Figure 3 illustrates the results. Figure 3(a) shows median grasp error as a function of trial number. This is the grasp error measured after reaching to the object, but before executing the GRASP controller. Before the 10th trial, the large median grasp errors indicate that SCHEMA STRUCTURED LEARNING had not yet discovered how to grasp the towel roll. This means that the GRASP controller must correct this poor configuration by displacing the contacts along the object surface toward a good grasp configuration. However, by the 10th or 15th trial, the robot has learned to select an instantiation of the REACH controller that minimizes moment residual errors.

Figure 3(b) illustrates the manipulator poses to which the robot learns to reach. This contour plot shows the probability of grasp success as a function of orientation (vertical axis) and position (horizontal axis) relative to the object. Both position and orientation are measured in the same way as with the REACH controllers in Section II: position is the distance of



(a) (b)

Fig. 3. Results from the eight grasping experiments. (a) shows median initial grasp controller error (moment residual error) as a function of trial. This is the quality of the grasp after executing the reach controller, but before the grasp controller. A high error indicates a poor grasp and a low error indicates a good grasp. Note that median error reaches its lowest point after approximately 10 trials. (b) is a contour plot that shows the learned grasp knowledge. The probability of grasp success is maximized when the manipulator reaches to a position halfway up the object (between 0.3 and 0.9) and an orientation almost perpendicular to the object major axis.

the contact centroid between the center and one end of the major axis; orientation is the average angle formed by the major axis and the line between a contact and the contact centroid. These results show that the robot learns that the probability of a successful grasp is maximized when it reaches to a position halfway up (between 0.3 and 0.9 of the distance from the object centroid along the major axis) the length of the object and at an orientation almost perpendicular (greater that 1.35 radians) to the object major axis. Intuitively, if the robot reaches too close to the center of the object, the palm of the hand collides with the object and the grasp fails. In addition, a three-fingered grasp is optimized when the manipulator orientation is approximately perpendicular to the major axis.

VI. Conclusion

This paper expands on the action schema approach to robot learning proposed in [6]. In this approach, the search for desired robot behavior is constrained by a generalized policy encoded by the action schema. This simplifies learning: instead of considering all possible behaviors, the robot must only consider instantiations of the generalized action schema policy. This paper defines the optimal policy instantiation as that which maximizes the probability of satisfying action schema transition specifications. In an approach reminiscent of Piaget's circular reaction, this paper proposes a sample-based algorithm, SCHEMA STRUCTURED LEARNING, whereby the robot repeatedly executes instantiations of the action schema policy in a search for optimal instantiations. This paper reports on experiments conducted on Dexter, the UMass bimanual humanoid, where SCHEMA STRUCTURED LEARNING learns how to reach and grasp an object. The results show that the system learns within an average of 10 trials how best to reach and grasp an object.

Acknowledgment

The authors would like to thank Emily Horrell for her help maintaining Dexter and running experiments. This work was supported by NASA grant NNJ05HB61A, ARO grant DAAD 19-03-R-0017, and NASA GSRP Fellowship NNJ04jf76H.

References

[1] M. Huber, "A hybrid architecture for adaptive robot control," Ph.D. dissertation, U. Massachusetts, 2000.
[2] R. Arkin, *Behavior-Based Robotics*. MIT Press, 1998.
[3] J. Piaget, *The Origins of Intelligence in Children*. Norton, NY, 1952.
[4] M. Arbib, "Schema theory," in *Encyclopedia of Artificial Intelligence (2nd Edition)*. Wiley-Interscience, 1992.
[5] G. Drescher, *Made-Up Minds: A Constructivist Approach to Artificial Intelligence*. MIT Press, 1991.
[6] R. Platt, A. H. Fagg, and R. A. Grupen, "Re-using schematic grasping policies," in *IEEE Int'l Conf. on Humanoid Robots*, 2005, pp. 141–147.
[7] J. Coelho, J. Piater, and R. Grupen, "Developing haptic and visual perceptual categories for reaching and grasping with a humanoid robot," *Robotics and Autonomous Systems Journal, special issue on Humanoid Robots*, vol. 37, no. 2-3, November 2001.
[8] J. Coelho and R. Grupen, "A control basis for learning multifingered grasps," *Journal of Robotic Systems*, 1997.
[9] R. Platt, A. H. Fagg, and R. A. Grupen, "Nullspace composition of control laws for grasping," in *IEEE Int'l Conf. on Intelligent Robots and Systems*, 2002.
[10] ——, "Manipulation gaits: Sequences of grasp control tasks," in *IEEE Int'l Conf. Robotics Automation*, 2004.
[11] R. Platt, A. Fagg, and R. Grupen, "Extending fingertip grasping to whole body grasping," in *IEEE Int'l Conference on Robotics and Automation*, 2003.