2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

Intrinsically Motivated Hierarchical Manipulation

Stephen Hart, Shiraj Sen, and Rod Grupen

Laboratory for Perceptual Robotics
Computer Science Department
University of Massachusetts Amherst
{shart,shiraj,grupen}@cs.umass.edu

Abstract— **We present a framework for the programming of manipulation behavior by means of an intrinsic reward function that encourages the building of deep control knowledge. We show how this framework can be used to teach new manipulation skills in a hierarchical and incremental fashion. We demonstrate the contributions of this paper on a humanoid robot through three incremental learning stages.**

I. Introduction

In this paper, we propose a new approach to programming robotic manipulation tasks. The approach builds upon the control basis framework for structuring learning tasks in robotics. We introduce a novel intrinsic reward function expressed in this framework. Our goal is to provide the learning agent with an intrinsic motivation to discover and organize control knowledge hierarchically. The "multi-modal imperative" (MMI) rewards the discovery of new affordances for control. It remains fixed over the lifetime of the learning agent and is the sole source of reward in our experiments.

After a discussion of related work, Section III provides an introduction to the control basis framework for constructing closed-loop controllers with embedded state information. The new intrinsic reward function is then introduced that rewards the discovery of *catalogs*, a cumulative memory structure that records affordances for predictable robot behavior. Section V shows how reinforcement learning techniques are used to build new control programs in training situations that severely restrict the range of states and actions that can be explored. These new programs can be used as abstract, temporally extended actions in subsequent programs. Section VI provides demonstrations of the staged learning process in which three hierarchically organized control programs are constructed.

II. Related Work

The work in this paper extends upon the "control basis" framework originally developed in the contributions of Huber [10], and Coelho [5], to provide new formalisms for hierarchy, abstraction, and development. We extend the control basis framework to allow a robot to acquire new skills through intrinsic drives and incremental learning stages.

Providing intrinsic drives for behavior acquisition and knowledge discovery have been well studied in the psychology literature beginning with Berlyne [4] who introduced key factors such as *novelty* and *curiosity*. Intrinsically rewarding mechanisms to motivate skill acquisition has also been applied in robot and reinforcement learning systems [3], [9].

Computational methods for incremental learning in robotic systems was proposed by Asada *et al.* [1] and have recently enjoyed much attention in the developmental robotics literature [15]. Staged learning has also provided means for grounded knowledge transfer as an agent learns increasingly complex skills [2], [13], [6]. Much of this work considers low-level sensorimotor mappings, whereas we focus on finding grounded *behavioral* affordances.

III. The Control Basis

The *control basis* provides a means of computing reference inputs to lower-level motor units. In our implementations, motor units are second order (PD) position and force controllers. The control basis is a combinatoric framework for constructing closed-loop controllers with a means of estimating state information that supports optimal control decisions.

A. Control Actions

The control basis is defined by three sets: potential functions Ω_ϕ, feedback signals Ω_σ, and motor parameters Ω_τ. While Ω_σ and Ω_τ are grounded in the robot's sensors and actuators. Ω_ϕ, in contrast, describes a set of potential functions that serve as primitive subtasks for integrated behaviorial programs.

Primitive actions in the control basis framework are closed-loop feedback controllers constructed by combining a potential function $\phi \in \Omega_\phi$, with a feedback signal $\sigma \in \Omega_\sigma$, and motor variables $\tau \in \Omega_\tau$. In any such configuration, $\phi(\sigma)$ is a scalar potential function (i.e., a *navigation* function [14]) defined to satisfy properties that guarantee asymptotic stability.

Examples of potential functions include fields that describe kinematic conditioning [8], harmonic functions for collision-free motion [7], and force closure functions [17]. In this paper, we will make extensive use of a simple quadratic function,

$$\phi_{\epsilon^T\epsilon}(\sigma) \quad = \quad \sigma^T\sigma, \tag{1}$$

where σ denotes an error signal determined from features in the feedback signals. Potential function $\phi_{\epsilon^T\epsilon}$ can be used to compute quadratic error functions in generic spaces (workspace, configuration space, force space, and/or torque spaces).

The sensitivity of the potential to changes in the value of motor variables is captured in the error Jacobian, $J =$

$\partial\phi(\sigma)/\partial\tau$. Reference inputs to lower-level motor units are computed by controllers $c(\phi, \sigma, \tau)$, such that

$$\Delta\tau = J^{\#}\phi(\sigma),$$

where $J^{\#}$ is the pseudoinverse of J [16].

Multi-objective control actions are constructed by combining control primitives. Concurrency is managed by projecting subordinate/inferior actions into the nullspace of superior actions.

$$\Delta\tau = J^{\#}_{sup}\phi_{sup} + \left[I - J^{\#}_{sup}J_{sup}\right] J^{\#}_{inf}\phi_{inf}. \quad (2)$$

This prioritized mapping assures that inferior control inputs do not destructively interfere with superior objectives. This approach can be extended to n-fold concurrency relations. In the following, we will use a shorthand for the nullspace projection that uses the "subject-to" operator "$\triangleleft$." The control expression $c_{inf} \triangleleft c_{sup}$—read, "$c_{inf}$ *subject-to* c_{sup}"—is shorthand for Equation 2.

The combinatorics of potentials Ω_ϕ, and resources Ω_σ and Ω_τ defines all primitive closed-loop actions $\mathcal{A}$ that the robot can employ. Previous work by Huber [12], [11] and Platt [17] addresses how to direct exploration using constraints on $\mathcal{A}$ and how such programs generalize to new contexts.

B. Controller State

The error dynamics (ϕ, $\dot{\phi}$) created when a controller interacts with the task domain supports a natural discrete abstraction of the underlying continuous state space [5]. In this paper, we will use a simple discrete state definition based on *convergence events*. In the following, we define a predicate $p(\phi, \dot{\phi})$ associated with a controller

$$p(\phi, \dot{\phi}) \quad = \quad \begin{cases} X & : \quad \phi \text{ state is unknown} \\ - & : \quad \phi \text{ has undefined reference} \\ 0 & : \quad \dot{\phi} < \epsilon_\tau \\ 1 & : \quad \dot{\phi} \geq \epsilon_\tau, \end{cases}$$

where ϵ_τ is a small negative constant that depends on effector resources τ. For asymptotically stable controllers, ϕ is positive definite and $\dot{\phi}$ is negative definite. The "$-$" condition means that no target stimuli is present in the feedback signal. Thus, a collection of n distinct primitive control actions forms a discrete state space $\mathcal{S} \equiv (p_1, \cdots, p_n)$.

IV. Intrinsic Reward

In this section, we introduce a novel *intrinsic* reward function that is used to create an inherent motivation in the learning agent to discover new control knowledge. This function rewards only control actions that rely on *direct* feedback from the external world.A reward occurs for a control action if that action causes a new transition from $p(\phi, \dot{\phi}) = 0$ to $p(\phi, \dot{\phi}) = 1$ for some resource configuration, (σ, τ). The goal is to discover as many of such interactions per task. For example, a mobility controller that achieves the goal state will reward the perceptual and motor behavior that leads to it, recognizing environments it can traverse in terms of previous observations. Likewise, a successful (converged) grasp controller, $c_g(\phi, \sigma, \tau)$, rewards actions that preceed $p_g(\phi, \dot{\phi}) = 1$, specifying the environmental conditions supporting "graspable" objects.

The simple reward function presented in Equation 3 has the properties we require in an intrinsic reward function.

$$r(t) = \|\mathcal{C}\| \left[(m_i \wedge (c_i(\phi, \sigma, \tau) \notin \mathcal{C})\right], \quad (3)$$

A control transition event at time t, m_i is defined to be a binary assertion that controller, c_i, causes the transition from $p(\phi, \dot{\phi}) \neq 1$ to $p(\phi, \dot{\phi}) = 1$. We also define a memory structure called a *catalog*, $\mathcal{C}$, that records affordances in terms of control configurations, $c_i(\phi, \sigma, \tau)$. It is a structure in memory that logs controllers that succeed in asserting m_i. When a new m_i is associated with other events comprising a particular catalog, but has not yet been recorded, the controller that precipitated the new event is added to the catalog and the motor policy is rewarded for making a new discovery. If it already exists in the catalog, the controller produces no reward. Moreover, a new control transition event, m_i, is rewarded by an amount that is proportional to the number of other controllable events in the catalog. Thus, the robot receives 1 unit of reward for seeing a novel stimulus, and 2 units of reward if it touches it, 3 if it can form a stable grasp, etc. We refer to this reward function as the "multi-modal imperative," or MMI. This intrinsic reward function encourages creating new behavior and deep catalogs around aspects of an environment that already exhibit some degree of coherence and simultaneous controllability.

In addition to creating commonsense knowledge about interactions with the environment, the reward function Equation 3 also motivates the construction of new temporally extended control programs that can assert new properties of value in the catalog. As we stated in the introduction, the demonstrations in this paper are designed to show that even very simple, highly constrained learning problems subject to this intrinsic reward function lead to hierarchies of control programs.

V. Hierarchical Programs

In this section, we show how sequential programs can be assembled out of control primitives expressed in the control basis framework. We construct a Markov Decision Process (MDP) consisting of states, $\mathcal{S}$, actions, $\mathcal{A}$, transition dynamics, $\mathcal{T}$, and reward, $\mathcal{R}$. A learning agent can estimate the *value*, $\Phi(s, a)$, of taking an action a in a state s in terms of the expected future reward using reinforcement learning (RL) techniques such as Q-learning [18]. Q-learning estimates the value function through trial-and-error experience using the update-rule:

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

where $\gamma \in [0, 1]$ is the discount rate and $\alpha \in [0, 1]$ is the learning rate. With sufficient experience, this estimate is guaranteed to converge to the optimal value Φ^*. The optimal policy, π, is a greedy ascent of the optimal value function. It maps states to actions by maximizing the expected sum of discounted future reward, such that $\pi(s) = \text{argmax}_{a_i}\Phi(s, a_i)$.

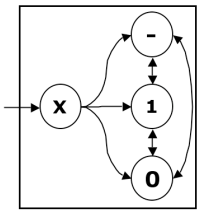


Fig. 1. This figure shows an iconic representation of a state transition when a sensorimotor program is invoked hierarchically.

Q-learning and the intrinsic reward described in Section IV discovers controllable interactions with the world. This policy can be re-used when the environment affords similar features and control events. Value functions provide a natural hierarchical generalization of potential functions—greedy ascent of the value function is an optimal strategy for discovering reward and absorbing states correspond to convergence events with $\dot{\Phi}$=0.

The basis for hierarchy in the control basis framework depends on the abstraction of sensorimotor programs, with all the internal state they require, in terms of a single discrete state variable as shown in Figure 1. Although a program can have a significant amount of internal structure, the hierarchical learning agent views this program as a single, temporally extended control action with three probabilistic outcomes.

VI. Staged Learning Experiments

We demonstrate three learning stages with Dexter (Figure 2), a research platform at the Laboratory for Perceptual Robotics at UMass Amherst. Feedback signals available in the

Fig. 2. The experimental Dexter platform for studying learning algorithms and control frameworks in bimanual dexterous manipulation.

Dexter platform consists of joint angle positions and velocities for 23 DOF, stereo visual feedback (hue, saturation, intensity), real-valued force feedback from fingertip contact sensors, and motor currents in arm actuators. All of these values are continuous in time with overlapping receptive fields. Contiguous homogeneous regions on the image plane are described in terms of the first and second moments of these distributions. In addition, derivative of Gaussian (dog) operators extract texture in spatial and temporal signals.

In this paper, we focus on demonstrating how a simple learning process and an intrinsic reward function give rise to hierarchical programs. We demonstrate a three stage, hierarchical learning process in which a new program can be "taught" to Dexter in each stage and used as a temporally extended action in subsequent stages. The robot is subject to a single, fixed reward function (Section IV) that rewards the discovery of affordances for controllable interactions with the world. Guided by resource constraints and training regimens, it learns an array of 3 interdependent behaviors that allow it to localize, reach to, hold, and inspect objects in its workspace. Each stage consists of 25 learning episodes in which Dexter uses Q-learning with ϵ-greedy exploration ($\alpha = 0.1$, $\gamma = 0.8$ and $\epsilon = 0.2$). Each episode ends when the current level of the hierarchy receives a new reward not in the catalog.

A. *Stage 1:* SaccadeTrack

Stage 1 is designed to program a simple skill for finding and tracking visual stimuli with Dexter. We do so under the simplest conceivable context; the robot looks in places where motions are expected and tracks motion cues if they are detected.

The program called SaccadeTrack is constructed by searching sequential and concurrent combinations of two actions $a_{saccade}$ and a_{track}. Allowing for nullspace compositions of these actions, the possible actions in stage 1 are $\mathcal{A}_1 = \{a_{saccade}, a_{track}, a_{saccade} \triangleleft a_{track}, a_{track} \triangleleft a_{saccade}\}$.

Saccade constructs a feedback error from two signals, the value of the joint angles in the stereo head, $\vec{\theta}_{head,act}$ and a reference posture for the head, $\vec{\theta}_{head,ref}$. Dexter's introduction to SaccadeTrack considers motion—non-zero velocity of a homogeneous region on the image plane—exclusively as a sensory cue. Moreover, *saccade* makes guesses about head postures where the image coordinates motion cues $\vec{u}_{motion} \in \mathcal{U}$ are likely to be detected on the center of the image plane $\vec{u}_0$. These postures are sampled from the distribution, $\vec{\theta}_{head,ref} \sim Pr(\vec{\theta}_{head} | \vec{u} = \vec{u}_0)$. The error in the feedback signal is thus, $\epsilon_{p(motion)} = (\vec{\theta}_{head,ref} - \vec{\theta}_{head,act})$, and the potential function is $\phi_{\epsilon^T \epsilon}$. The effector variables, in this case, are positions in the configuration space of the head, $\vec{\theta}_{head}$, therefore, the saccade controller can be written

$$a_{saccade} \triangleq c(\phi_{\epsilon^T \epsilon}, \epsilon_{p(motion)}, \vec{\theta}_{head}).$$

Saccade orients the head to postures where the target motion is likely to be found—it increases the probability that motion cues will be found on both image planes of the stereo head.

Track is a closed-loop controller that pursues a motion cue by changing the reference head posture, $\vec{\theta}_{head}$. Dexter has a pan-tilt head with left and right cameras fixed along parallel gazes. The goal is to keep the coordinate of the *motion* cue $\vec{u}_{motion} = (u_l, v_l, u_r, v_r)$ at the origin (image center) $\vec{u}_0$ on both images (left and right) simultaneously, the error is $\epsilon_{motion} = (\vec{u}_0 - \vec{u}_{motion})$, and the potential function is defined by $\phi_{\epsilon^T \epsilon}$. Therefore, the track controller is written:

$$a_{track} \triangleq c(\phi_{\epsilon^T \epsilon}, \epsilon_{motion}, \vec{\theta}_{head}).$$

During training, the experimenter presents objects held in their hand roughly 30% of the time. The SACCADETRACK program (Figure 3), is acquired by learning to achieve the intrinsic reward when a_1 transitions from $p_{track} = 0$ to $p_{track} = 1$. Controller $a_{saccade}$ is an unrewarding control transition since it depends soley on empirical models of $Pr(\vec{\theta}_{head}|\vec{u} = \vec{u}_{motion})$ and not on direct feedback from the environment. The state space is $\mathcal{S}_1 \equiv (p_{saccade}, p_{track})$. The

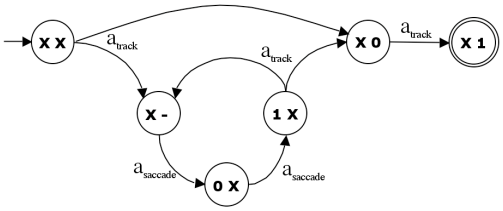
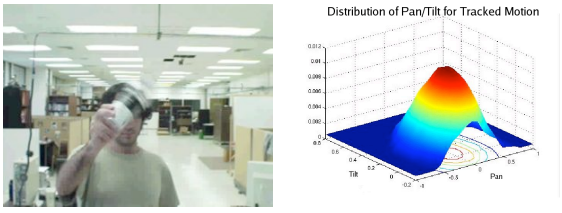


Fig. 3. A SACCADETRACK policy learned on Dexter in stage 1, characterized by the state space $\mathcal{S} = \{p_{saccade}, p_{track}\}$. The policy employs a_{track} first, and $a_{saccade}$ is chosen only when no stimuli is immediately present.

program begins in state $s = (XX)$ where policy SACCADETRACK chooses action a_{track}. If the target motion stimulus is absent, the state transitions to $(X-)$ and SACCADETRACK enters a loop that invokes $a_{saccade}$ until a stimulus is detected. If however, the motion cue is detected, the resulting state is $(X0)$ and if a_{track} completes the transition to state $(X1)$, the learning agent is rewarded. Whenever the motion cue is present in both images, a stereo triangulation mapping produces the Cartesian location of the stimuli on the pair of image planes. This Cartesian data stream is an abstract sensor signal that can be used by other control processes.

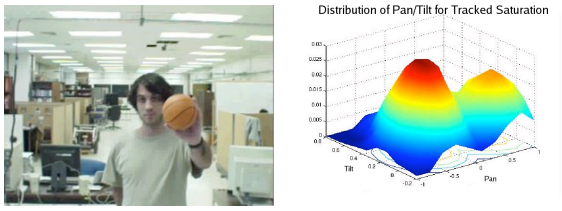
After 25 trials, the posterior estimate for $Pr(\vec{\theta}_{head}|\vec{u} = \vec{u}_{motion})$ (Figure 4(b)) has evolved from a uniform distribution to a unimodal peak describing how the experimenter presented movement stimuli to the learning robot (Figure 4(a)).



(a) (b)

Fig. 4. Panel (a) shows Dexter's left camera view while tracking motion during a typical programming trial, and (b) shows the non-parametric distribution of pan/tilt configurations learned for motion cues after 25 training episodes. The single peak corresponds to the place where the experimenter presented motion cues to the robot during the acquisition of SACCADETRACK.

The robot can reapply SACCADETRACK behavior using features other than motion, like the discrete hues, saturations, intensities, textures, available on the Dexter platform. It can do so without re-learning the SACCADETRACK strategy, and instead, focus exclusively on the feature-dependent part of the task. Figure 5 shows the presentation of stimuli and the corresponding probabilistic distribution of this range of saturated pixels in Dexter's environment.



(a) (b)

Fig. 5. Panel (a) shows the image from Dexter's left camera while executing SACCADETRACK on a discrete range of highly saturated pixels. Panel (b) shows the non-parametric distributions after 25 training episodes summarizing the pan/tilt configurations where Dexter expects to observe this range of saturation in the visual feedback.

The robot learns the SACCADETRACK strategy to achieve intrinsic rewards associated with 0-1 transitions on p_{track} in the motion training context. Dexter is likewise rewarded when it discovers that the transition dynamics for SACCADETRACK apply to other features as well and adds them to a catalog describing the environment and the experimenter's behavior. Over time, the robot optimizes SACCADETRACK across many features.

B. *Stage 2:* REACHGRAB

In the second stage, a REACHGRAB program is constructed that reaches to a Cartesian location and attempts to grasp an object at that location. The target locations for the *reach* action could be based on probability distributions like $Pr(\vec{f}_{tactile}|\vec{x}_{hand})$ (the likelihood of making tactile contact given the Cartesian position of the robot's hand) in a manner analogous to the *saccade* action. In this case, Dexter would have better than random guidance to achieve tactile responses in the absence of other sensory information. However, the learning problem posed in this paper always has access to the Cartesian location generated by SACCADETRACK, which proves to be a potent cue for placing the robot's hands on objects in the environment. Therefore, REACHGRAB can explore actions that re-use prior knowledge in the form of SACCADETRACK. In addition, several more primitives are allowed in the set of available actions.

Reach constructs a feedback error between the Cartesian location of left hand and the location determined by SACCADETRACK, $\epsilon_x = (\vec{x}_{object} - \vec{x}_{hand})$, where $\vec{x}_{hand} \in R^3$ is determined by the forward kinematic relationship for the left arm derived from direct sensor measurements of $\vec{\theta}_{arm}$. The reach controller submits a stream of position references to the underlying motor units. Therefore, the reach controller can be written

$$a_{reach} \triangleq c(\phi_{\epsilon^T\epsilon}, \epsilon_x, \vec{\theta}_{arm}).$$

a_{reach} provides a control input based on an indirect (kinematic) transformation, and is thus unrewarding by the intrinsic motivator.

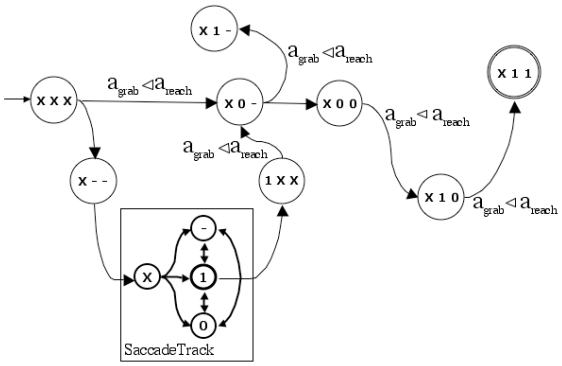


Fig. 6. This diagram shows the policy learned for REACHGRAB after 25 episodes. The state space is $s = (p_{SaccadeTrack}, p_{reach}, p_{grab})$. The hierarchical use of SACCADETRACK is indicated by the abstract transition icon introduced in Figure 1.

(a) (b)

Fig. 7. Panel (a) shows Dexter reaching to a highly saturated object and panel (b) shows Dexter holding that object with the force controllers in the hand.

Grab is a very simple control task to generate grasps. More sophisticated grasp controllers have been assembled using the control basis [17]. However, in this paper, our focus is on intrinsic motivation for hierarchical behavior, not on grasping control. In our "grab" controller, the hand is held in a fixed posture with fingers extended. Action *grab* is a closed-loop force controller that produces reward by controlling finger flexion in direct response to pre-defined reference forces on the 3 fingertip load cells of the left hand. The error signal is written $\epsilon_f = (\vec{f}_{ref} - \vec{f}_{hand})$ and the controller is defined:

$$a_{grab} \triangleq c(\phi_{\epsilon^T \epsilon}, \epsilon_f, \vec{\theta}_{hand})$$

The set of possible actions for stage 2, including nullspace compositions of the primitive controllers, is defined by $\mathcal{A}_2 = \{a_{reach}, a_{grab}, a_{reach} \triangleleft a_{grab}, a_{grab} \triangleleft a_{reach}, \text{SACCADETRACK}\}$. The state space is $\mathcal{S}_2 \equiv (p_{SaccadeTrack}, p_{reach}, p_{grab})$.

The learned REACHGRAB policy is shown in Figure 6. During 25% of these learning episodes, the experimenter presented graspable objects with highly saturated colors to Dexter. The presentation was either handing the object to Dexter or placing it on the table in front of the robot. Consequently, Dexter learned to reach out and grasp the objects handed to it, as seen in Figures 7(a) and 7(b), or to employ SACCADETRACK to find and grasp the object. Sometimes, Dexter was observed reaching to competing visual stimuli in the room (i.e., the window) and was unable to complete a successful grasp.

C. Stage 3: VISUALINSPECT

In stage 3, one additional primitive controller is added to the control resources available to the robot. We define a *localizability* controller to help control Dexter's stereo perspective on objects. The addition of this primitive allows for deeper behavioral catalogs because it can increase visual acuity and provide high-frequency features to visual tracking controllers [8].

Localizability is a quality of the geometry of a stereo pair. The geometry of the cameras and a viewed subject determines how precisely the Cartesian location of that subject can be estimated. The triangulation equations are linearized by differentiation to create a stereo imaging Jacobian J that can be used to analyze the sensitivity of triangulation results with respect to uncertainty in the location of features on the image plane [19].

A potential field in Cartesian space is defined by computing the scalar conditioning metric of the stereo triangulation Jacobian, $\phi_{cond}(J) = \sqrt{det(JJ^T)}$. When used as a control objective with a manipulator, the gradient of this potential field can be used to adjust the geometry and improve the visual acuity of grasped objects.

In this stage, we define a control action a_{cond} that is unrewarding, but that improves localizability according to this metric for highly-saturated objects that are held by the robot.

$$a_{cond} \triangleq c(\phi_{cond}, J, \vec{\theta}_{arm}).$$

This potential function operates in Cartesian space—its input and output are in workspace coordinates. Therefore, the position references to the underlying motor units are derived from $\Delta\vec{\theta}_{arm} \approx J_m^{\#}\phi_{cond}$, where J_m is the manipulator Jacobian that transforms displacements in workspace into displacments in configuration space.

Track is invoked once again, as in stage 1, but this time with a different visual feature—homogeneous regions of blue hue $\vec{u}_{blue}$. This features will be the target for a new tracking action,

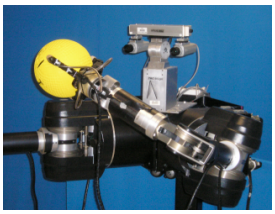
$$a_{track(blue)} \triangleq c(\phi_{\epsilon^T \epsilon}, \epsilon_{blue}, \vec{\theta}_{head}),$$

where $\epsilon_{blue} = (\vec{u}_0 - \vec{u}_{blue})$.

Allowing for nullspace compositions of primitive controllers, the possible actions for this stage are $\mathcal{A}_3 = \{a_{track(blue)}, a_{cond}, a_{track(blue)} \triangleleft a_{cond}, a_{cond} \triangleleft a_{track(blue)}, ReachGrab\}$. The state space is $\mathcal{S}_3 \equiv (p_{ReachGrab}, p_{cond}, p_{track(blue)})$.

Dexter uses the fixed intrinsic reward function in the context of $\mathcal{A}_3$ to reward a 0-1 transition in the second (blue) feature associated with this object, but not previously in the catalog $\mathcal{C}$. Figure 9 shows the policy learned for this program. 25% of the time the blue feature could be tracked immediately, when the object was held close to the robot's cameras, and leading to the state $(XX1)$. The rest of the time, however, the small blue feature patch was not sufficiently visible by

(a) (b)



(c)

Fig. 8. (a) and (b) show Dexter's left camera image before and after the execution of the control law $a_{cond} \triangleleft a_{track(blue)}$. (c) shows Dexter after the execution of that law.

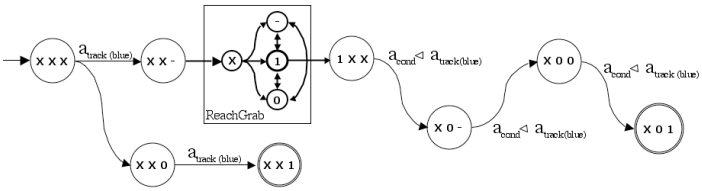


Fig. 9. This diagram shows the policy learned for VISUALINSPECT after 50 episodes. The state space is $\mathcal{S} = (p_{ReachGrab}, p_{cond}, p_{track(blue)})$.

the robot's cameras at the initial object location, as seen in Dexter's field of view in Figure 8(a). During the course of the learning episodes, however, Dexter learns to reach out and grasp the ball using the REACHGRAB program learned in stage 2—and then to visually condition the saturated ROI of the ball by executing the a_{cond} action, bringing the ball towards the location seen in Figure 8(c). Note, however, that the learned policy has the conditioning action run subject to $a_{track(blue)}$, such that when the blue feature becomes visible in state $(X00)$, it can be tracked and more reward can be attained. This often happens in state $(X01)$, before the conditioning controller completes.

VII. DISCUSSION

In this paper, we demonstrate a framework in which a fixed intrinsic reward function and a sequence of learning contexts gives rise to hierarchical control programs that represent general purpose skills for use by a robot. The intrinsic reward function motivates the discovery of behavior and affordances for that behavior. Learning contexts are defined by the range of robot resources available for autonomous exploration during each stage of programming. We developed programs called SACCADETRACK, REACHGRAB and VISUALINSPECT to provide knowledge that can be re-used in other manipulation tasks.

In future work, we plan on generalizing this method to many more channels of sensory feedback available in Dexter and to extend the hierarchy of behavior to new skills such as PICKANDPLACE and SORT, as well as skills that require knowledge and behavior for multi-body relationships such as STACK, INSERT, and ASSEMBLE.

VIII. ACKNOWLEDGMENTS

This material is based upon work supported under Grants ARO W911NF-05-1-0396, and NASA NNX07AD60A. The work is also supported by the NASA Graduate Student Research Program fellowship NNJ05JG73H, We would like to thank Shichao Ou and Emily Horrell for their help and support.

REFERENCES

[1] M. Asada, K. MacDorman, H. Ishiguro, and Y. Kuniyoshi. Cognitive developmental robotics as a new paradigm for the design of humanoid robots. *Robotics and Autonomous Systems*, 37(2):185–193, 2001.

[2] M. Asadi, V. N. Papudesi, and M. Huber. Learning skill and representation hierarchies for effective control knowledge transfer. In *ICML 2005 Workshop on Structural Knowledge Transfer for Machine Learning*, Pittsburgh, PA, 2006.

[3] A. Barto, S. Singh, and N. Chentanez. Intrinsically motivated learning of hierarchical collections of skills. In *Proceedings of the International Conference on Development and Learning (ICDL)*, LaJolla, CA, 2004.

[4] D. E. Berlyne. *Conflict, Arousal, and Curiosity*. McGraw-Hill, 1960.

[5] J. A. Coelho. *Multifingered Grasping: Grasp Reflexes and Control Context*. PhD thesis, Department of Computer Science, University of Massachusetts Amherst, 2001.

[6] P. R. Cohen, Y. Chang, and C. T. Morrison. Learning and transferring action schemas. In *Proceedings of the 2007 International Joint Conference on Artificial Intelligence*, Hyderabad, India, 2007.

[7] C.I. Connolly and R.A. Grupen. Nonholonomic path planning using harmonic functions. Technical Report 94-50, University of Massachusetts, Amherst, 1994.

[8] S. Hart and R. Grupen. Natural task decomposition with intrinsic potential fields. In *Proceedings of the 2007 International Conference on Intelligent Robots and Systems (IROS)*, San Diego, California, 2007.

[9] X. Huang and J. Weng. Novelty and reinforcement learning in the value system of developmental robots. In *Proceedings of the 2nd International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems*, 2002.

[10] M. Huber. *A Hybrid Architecture for Adaptive Robot Control*. PhD thesis, Department of Computer Science, University of Massachusetts Amherst, 2000.

[11] M. Huber and R. Grupen. Learning to coordinate controllers - reinforcement learning on a control basis. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Nagoya, JP, August 1997. IJCAI.

[12] M. Huber, W. MacDonald, and R. Grupen. A control basis for multilegged walking. In *Proceedings of the Conference on Robotics and Automation*, Minneapolis, MN, April 1996. IEEE.

[13] F. Kaplan and V. V. Hafner. Mapping the space of skills: An approach for comparing embodied sensorimotor organization. In *Proceedings of the 4th IEEE International Conference on Development and Learning*, 2005.

[14] D.E. Koditschek and E. Rimon. Robot navigation functions on manifolds with boundary. *Advances in Applied Mathematics*, 11(4):412–442, 1990.

[15] M. Lungarella, G. Metta, R. Pfeifer, and G. Sandini. Developmental robotics: A survey. *Connection Science*, 15(4):151–190, 2003.

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

[17] R. Platt, A. H. Fagg, and R Grupen. Nullspace composition of control laws for grasping. In *International Conference on Intelligent Robots and Systems (IROS)*, Laussane, Switzerland, 2002. IEEE/RSJ.

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

[19] S. Uppala, D. Karuppiah, M. Brewer, S. Ravela, and R. Grupen. On viewpoint control. In *IEEE Conference on Robotics and Automation*, Washington, DC, May 2002. IEEE.