A Framework For Humanoid Control and Intelligence

Robert Platt[1], Oliver Brock[1], Andrew H. Fagg[1], Deepak Karupiah[1], Michael Rosenstein[1], Jefferson Coelho[1], Manfred Huber[2], Justus Piater[3], David Wheeler[1], and Roderic Grupen[1]

[1] Laboratory for Perceptual Robotics
University of Massachusetts, Amherst, MA 01003, USA
[2] Department of Computer Science and Engineering
University of Texas, Arlington, TX 76019, USA
[3] Dept. of Electrical Engineering and Computer Science
Institut Montefiore (B28), I15
Universit de Lige, 4000 Lige Sart-Tilman, Belgium

Abstract. One of the goals of humanoid robotics research is the development of a humanoid capable of performing useful tasks in unknown or unpredictable environments. To address the complexities of such tasks, the robot must continually accumulate and utilize new control and perceptual knowledge. In this paper, we present a control framework whereby control and perceptual knowledge are used to learn robot control policies at different levels of abstraction. We show how task-relevant perceptual features can be discovered that make better control policies possible. We also explore how trajectories of closed-loop control policies can provide uniquely relevant state information. The approach presented in this paper is illustrated with several case studies on actual robot systems.

1 Introduction

One key goal of humanoid robotics research is to design human-like robots capable of autonomously performing a wide range of tasks in open domains. This objective requires a design that exploits redundancies inherent in the mechanism, addresses environmental uncertainty and stochasticity, and makes appropriate sensor and actuation abstractions. These abstractions should express human-specified, task-oriented programs and support the autonomous planning and learning of such programs.

With their large number of sensors and mechanical degrees of freedom, humanoid robots afford great redundancy in the way that tasks may be accomplished. Not only is there redundancy in the number of distinct choices that the mechanism has in solving a task or subtask (e.g., grasp an object with either the left or right hand), but within one of these distinct choices, there is often one or more excess mechanical degrees of freedom. Mechanical redundancy can be used to satisfy additional objectives (such as optimizing posture for energy utilization) or to allow multiple subtasks to be performed simultaneously (such as picking one object with the left hand while approaching a second object with right hand). One explicit goal in the design of our architecture is to exploit these redundancies so as to provide a set of robust control actions that allow flexibility in the representation of high-level programs.

Open environments require a robot to plan and learn under novel conditions. This must be done in a manner that ensures the safety of the mechanism and surrounding environment, and allows model estimation and learning to occur within a feasible amount of time. The artificial intelligence community has long acknowledged the importance of action and state abstraction in reducing the depth of an action plan (or control policy) search tree, and hence the time required to discover a feasible plan [25, 20]. In addition, such abstractions provide a context in which one may generalize plans to situations different from those in which the plans were originally constructed. Our architecture employs either hand-tuned or learned closed-loop control policies as temporally-extended actions that become single units for planning and learning at a higher level of abstraction [23, 13, 34, 40]. Our goal is to provide state and action abstractions that are an appropriately expressive language for user-level programming and for autonomous and semi-autonomous planning and learning.

Our approach is: 1) to provide a hierarchical architecture for expressing perception and control, 2) to use closed-loop controllers defined over continuous state and action spaces as a set of low-level, primitive control actions, and 3) to construct discrete, abstract state and action representations upon which higher-level control policies may be expressed. The abstract representation of state is initially defined in terms of controller-derived events, but as the robot interacts with environment, this representation grows as a function of the utility of the augmented state representation in performing some task. Each level of control policy abstraction hides implementation details of lower levels and provides a level of robustness in dealing with perturbations and uncertainties.

In this paper, we describe the control framework (section 2), illustrate several elements of the low-level *control basis* (section 3), show an example of how the *perceptual basis* may be augmented as a function of the robot's experience in the world (section 4), and illustrate how the different aspects of the framework interact through several case studies (section 5). Aspects of this control framework have been used with a variety of robotic platforms including walking machines, teams of mobile robots, several arm/hand systems, and the UMass Humanoid (figure **??**).

2 The Control Framework

Humanoid robots must interface with open environments in a robust and flexible manner. Closed-loop control policies are a necessary ingredient for obtaining this goal because they are applicable in stochastic environments and are well-suited to multi-objective control. A variety of researchers have suggested that closed-loop policies are appropriate as a fundamental unit of robot control [17, 14, 19, 23, 13]. Control policies as actions in and of themselves represent temporally-extended abstractions that can be leveraged to construct compact, task-oriented policies either through an explicit programming process or through one that relies on some form of machine learning.

Our control framework is outlined in figure 2 and combines closed-loop control primitives and perception through a hierarchy of abstraction actions. These closed-loop behavioral primitives work in low-dimensional operational spaces (as do behavior-based approaches in general), but they also transform a continuous state space into a set of discrete equilibria. To enumerate the range of possible concurrent control situations that can be considered from some initial state, we employ a hybrid Discrete Event Dynamic System (DEDS) framework. In this formalism [26, 35, 38], the state of the underlying system is assumed to evolve with the occurrence of a set of discrete events, some subset of which are controllable.

The perceptual operators represent both raw and abstracted sensory inputs. The abstractions hide details of the perceptual stream, allowing for appropriate generalization across many distinct sensory inputs. In addition, discrete abstractions are presented to the DEDS model as a way of augmenting the controller state representation. A task-level control policy for a given discrete system model can then be expressed in terms of a discrete state space (as captured by the DEDS model) and a discrete action space (as captured by the set of available closed-loop primitive controllers and the available perceptual operators). It is hoped that policy formation in this space is higher performance and is likely to be more robust by virtue of the closed-loop control primitives.

2.1 A Control Basis

In order to function in stochastic environments, control primitives must be robust to uncertainty and noise. This includes weak or non-functional actuators and noisy or incorrect data concerning the external environment. Closed-loop control is a well-studied mechanism for rejecting noise in these environments. In this approach, an error function formalizing the control objective is continuously evaluated and used to update control signals in a way that reduces error. Closed-loop control laws are robust to stochastic environmental perturbations because in the controllability region, the control law drives the system back toward the basin of attraction. Arrival to a region of configuration space where the control law error function is locally minimized represents a convergence event from the perspective of the DEDS-level model.

Multi-objective control is another indispensable component of successful robot function. For example, a robot capable of dexterous manipulation must be able to reach to and grasp an object while simultaneously avoiding obstacles. Closed-loop control laws can be concurrently combined to solve multi-objective problems safely and effectively. By characterizing a control objective using a scalar potential function over the configuration space, it is possible to determine, at every point, the control dimensions that are orthogonal to the steepest descent along the potential function. This orthogonal subspace (the nullspace of the control law) is an explicit description of the redundant degrees-of-freedom (DOFs) with

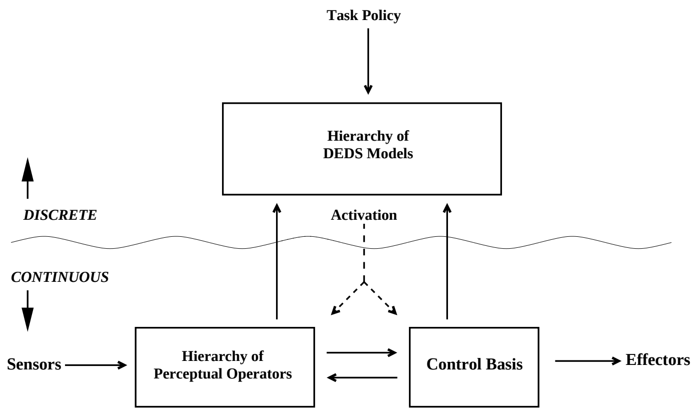

Fig. 2. The control framework. Solid lines indicate information flow and dotted lines indicate flow of control.

respect to the control law. Once identified, these redundant task dimensions can be used in service to other task objectives. In the case of grasping, for example, a control law specifies a potential function over the space of possible hand/arm configurations relative to an object such that a minimum exists in quality grasp configurations [7]. Potential field methods in motion planning impose a navigation function that, when projected into the nullspace of the grasp control law, can simultaneously drive the control point away from obstacles and toward the goal [8].

Our control framework formalizes this relationship using the "Subject To" constraint. Let Φ_1 and Φ_2 be two control laws that attempt to descend continuous, scalar-valued potential surfaces ϕ_1 and ϕ_2 respectively. We say that the composite controller $\Phi_C = \Phi_2 \triangleleft \Phi_1$ executes Φ_2 *subject to* Φ_1 when Φ_2 executes in the nullspace of ϕ_1, i.e., when $\nabla\phi_C = \nabla\phi_1 + \nabla\phi_2 N(\phi_1)$ where $N(\phi_1)$ is the nullspace of potential function ϕ_1. In this expression, the composite controller Φ_C descends the ϕ_1 potential function as a first priority. As a secondary priority, Φ_C also descends ϕ_2 when it does not interfere with the primary objective ϕ_1.

The *control basis* is described by a set of potential functions, a set of sensory inputs, and a set of effectors. In the general case, an element of the control basis is a "subject-to" ordered set of admissable combinations of potential functions, sensory inputs, and effectors.

2.2 A Perceptual Basis

Sensory feedback is an integral part of closed-loop control policies. In our framework, all sensory information is processed through the perceptual basis, the outputs of which are used both as inputs to elements of the control basis and as a discrete state representation for the DEDS-level models. The perceptual basis is a hierarchically organized set of operators. Each perceptual operator can be applied to different sensory sources. For example, an operator that convolves an input with a Laplacian of Gaussian basis function might be used to detect "blobs" in the currently-available visual and audio streams. Other possible perceptual operators detect object edges, texture, scale, orientation, objects [27, 28], or even human faces [42].

Actions from the perspective of the DEDS-layer models are implemented as *activations* of compatible control and perceptual basis elements. An individual element of the control basis may receive input from one of many alternative perceptual basis elements. For example, an object tracking controller may be altered to become a face-tracking controller through only a change in the selected perceptual basis element. This approach allows one to make explicit decisions about how computational resources are to be used at any instant in time. Moreover, it helps to focus the "attention" of policies defined at the DEDS-layer onto components of the incoming sensory stream that are particularly relevant to the task at hand.

Notice in figure 2 the flow of information from the control basis to the perceptual basis. This enables controller state to be treated on equal footing as the sensory inputs by elements of the perceptual basis. In particular, this is important for the construction of perceptual operators that examine the trajectory of controller evolution as a way of extracting *interaction-based state* representations.

The perceptual operators present discrete abstractions of sensory state to the DEDS-level models. The internal transformation from a continuous to a discrete representation is most often accomplished by thresholding the response of the perceptual operator to a particular input. For example, a perceptual operator might recognize the degree to which a visual input matches a particular shape feature, and indicate to the DEDS-level model whether or not this particular feature was observed.

2.3 A Hierarchy of DEDS Control Policies

Robot control is often regarded as a difficult decision making problem because it usually involves continuous state and action variables. The control framework described above makes this problem easier by providing discrete control actions and extracting discrete state information. States in our DEDS-level models capture the status of control and perceptual basis elements [12]. The DEDS framework is particularly useful in that it makes it possible to incorporate a variety of different DEDS-level policy formation mechanisms. In general, a policy is a probabilistic mapping from state to action. We are interested in at least two important types of policies: hand-crafted policies and autonomously learned policies. The DEDS framework makes it easy to combine the two. For each set of states, a human designer may specify a set of safe or acceptable actions. Autonomous learning can subsequently take over and optimize the policy within the space of safe alternatives. For the purposes of learning, we usually model the pruned DEDS model as an semi-Markov decision process (SMDP) for which a variety of learning methods, including reinforcement learning, are applicable [12].

We organize DEDS control into a hierarchical framework. For example, a DEDS-level control policy for correctly localizing, reaching, and grasping an object is a low-level policy because is has a simple policy based on the localize,

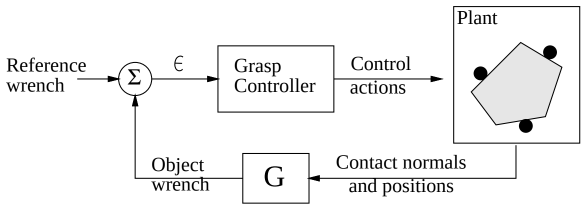

Fig. 3. Grasp synthesis as a control problem. The "G" box indicates that contact position and normal information is used to calculate net grasp wrench. The net wrench is compared with the reference wrench and appropriate contact displacements are generated.

reach, and grasp control basis elements. Once acquired, such a policy can become an action in its own right from the perspective of the next DEDS layer. For example, building a structure with blocks is a higher-level policy because it can be simply represented using localize/reach/grasp control policies as primitives.

3 Elements of the Control Basis

The control framework described in the previous section generates complex behavior by composing robust, parameterizable closed-loop controllers. The collection of controllers available within the framework is referred to as the control basis. The control basis should consist of flexible, yet expressive controllers that can contribute behavioral elements to a wide variety of high-level tasks. In this section we present elements of the control basis we use for grasping and gross motion generation. We also discuss how interactions between robot and human can be exploited to learn elements of the control basis.

3.1 Grasping

We approach grasping as a control problem [7]. Grasp controllers displace contacts on the surface of an object with unknown geometry so as to descend grasp error functions. At a physical level, the grasp controller executes a series of regrasps. In each control step, the fingers close until they lightly make contact with the object. The grasp controller uses tactile sensor information to compute controller error and error gradient with respect to contact position. After making light contact with the object, the controller lifts the contacts off the object and displaces them by a small amount calculated using a wrench-based error function. This process is depicted in Figure 3.

Grasping is fundamentally a multi-objective problem. The goal of grasping an object must be balanced with task-level requirements concerning how the object should be grasped and the kinematic optimizations of the hand and arm. A control-based approach to grasping can easily and intuitively incorporate multiple requirements like this by concurrently combining controllers. In the following two subsections, we will briefly describe controllers used in grasping and how they can safely be combined to function concurrently.

Control Laws for Grasping The grasp controller approach is based on an error function that has minima in configurations where the forces that would be applied by frictionless contacts result in a low net wrench. Several researchers including Ponce [33] have connected low net-wrench configurations and force closure. Net zero wrench applied by at least three frictionless contacts is a sufficient condition for force closure in an environment with a strictly positive coefficient of friction [33].

The grasp controller error synthesizes three manipulation control laws: a force-based contact position control law, a moment-based contact position control law, and a kinematic conditioning control law. The force-based contact position control law (Φ_{force}) is a potential function that has equilibria in configurations where frictionless contacts exert the reference net force. Similarly, the moment-based contact position control law (Φ_{moment}) has equilibria in configurations where frictionless contacts exert zero net moment. Finally, the kinematic conditioning configuration control law ($\Phi_{kinematic}$) has equilibria in configurations where the major axis of the finger's velocity ellipsoid is aligned with the contact normal. This optimizes the manipulator for controlled displacements tangent to the object surface.

The Subject-To Approach to Combining Control Laws Grasp control laws are combined concurrently by projecting some control laws into the nullspace of others:

$$\Phi_{kinematic} \triangleleft \Phi_{moment} \triangleleft \Phi_{force}.$$

This expression should read: $\Phi_{kinematic}$ subject to Φ_{moment} subject to Φ_{force} [32]. The "subject to" constraint is shorthand for a projection of one control law into the nullspace of another. The controller written above will reconfigure the manipulator to try to minimize net force as a first priority. If possible, it will also try to minimize net moment. Finally, it will optimize the kinematic configuration with respect to the object without disrupting the first two objectives.

The Subject-To approach should be contrasted with a direct combination of control laws. Approaches which simply superimpose controllers onto each other cannot characterize the behavior of the composite controller very well. If two control laws have opposite objectives in configuration space, they could cancel each other and no behavior (or incorrect behavior) would result. In contrast, the nullspace approach ensures that one control law is maximally effective while others participate subject to the first.

Whole Body Grasping While robotic grasping research typically assumes that fingertips alone will be used to grasp objects, non-fingertip contacts are also possible. For example, potential contact points may exist on the palm, the sides of the arm, or even the humanoid chest. We use the term "whole body grasp" to refer to grasps that depend on contacts on arbitrary hand and body surfaces [31].

The grasp controller approach described in this section can be used to form "whole body grasps" as well as grasps with the more conventional fingertip surfaces. In particular, we have applied these techniques to enveloping grasping with the NASA/JSC Robonaut hand and arm. By using contacts on the palm of the hand as input to a set of grasp controllers, we have used this approach to synthesize enveloping grasps on a simulation of Robonaut. Elements of this approach have also been used in Robonaut hardware implmentations.

The ability to parameterize grasp controllers with arbitrary contact surfaces greatly extends the number of possible grasp controller instantiations. Re-parameterizing the grasp controller with different resources is an example of how the control basis makes it possible to transfer control knowledge from one domain to another.

3.2 Gross Motion Generation

Dynamic collision avoidance is essential for many high-level tasks. For example, grasping and manipulation tasks require the robot to reach out with its end-effector in order to grasp an object or move an object into a particular configuration. During such a motion, collision avoidance has to be ensured for the entire manipulator. This section introduces elements of the control basis that provide the capability of real-time motion generation and collision avoidance for robots with many degrees of freedom, such as humanoid robots. By combining these gross motion controllers with other elements of the control basis, such as the grasping controller presented above, task-related behavior can be combined with collision avoidance.

Real-Time Path Modification The computational complexity of robot motion planning methods [1, 16, 21] found in the literature is generally too high to allow motion generation in high-dimensional configuration spaces for reliable movement in dynamic environments. Reactive motion generation methods, on the other hand, cannot address global constraints imposed by the task [17], as they are susceptible to local minima. We have developed a motion generation method that combines the global properties of motion planners with the reactive properties of local methods, while at the same time possessing the computational efficiency to be applied to kinematically complex robotic systems, such as humanoid robots.

The elastic strip framework [3] permits the real-time modification of a previously planned path in high-dimensional configuration spaces. The modification takes into account a variety of constraints imposed on the motion. These constraints can be imposed to avoid collision with moving or stationary obstacles, they can be the consequence of physical limitations of the mechanism (e.g., actuation constraints, maintaining balance), they might be a function of the task performed with the end-effectors, or they can simply express a preferred posture

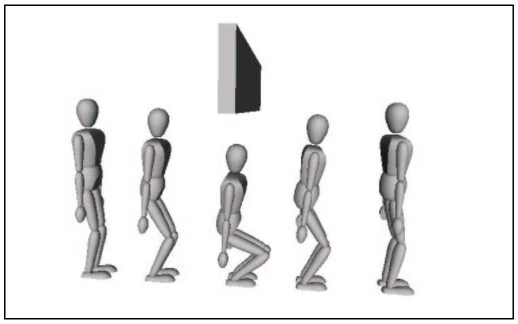

Fig. 4. Incremental modification of humanoid motion considering obstacle avoidance, balance constraints, and preferred posture.

to minimize power consumption or to make the motion appear human-like. The elastic strip framework can thus be viewed as an augmentation to conventional path planners, providing them with the the capability to execute sophisticated, task-oriented motion for mechanisms with many degrees of freedom in dynamic environments. Substantial changes in the environment, however, can invalidate the motion and prevent the incremental path modification procedure from maintaining the constraints. In such a case a global motion planner has to be invoked to determine a new path that satisfies all required constraints.

Starting with the path generated by a global motion planner to bring the end-effector into a position determined by the grasping controller, for example, the elastic strip framework augments the representation of that path with an approximation of the free workspace surrounding that path. This workspace volume can be viewed as a tunnel through which the robot is moving. Since the volume represents free space, any incremental modification of the path entirely containing the robot's motion inside that volume is guaranteed to be free of collision. The elastic strip framework applies efficient control methods [18] to select a modification consistent with the imposed constraints. Furthermore, it allows for the automatic suspension of desirable, but not critical constraints when they conflict with constraints whose violation would lead to catastrophic failure [3]. Figure 4 shows five snapshots of a humanoid figure with 34 degrees of freedom. The initial trajectory is modified in real-time during execution of the motion while observing three different constraints: the obstacle, which is being lowered during the motion, is avoided, the overall balance of the mechanism is maintained, and among all the possible motions conforming with the previous requirements, those that appear human-like are preferred. Note that the complexity of bipedal walking is being ignored in this experiment.

Real-Time Path Planning Controllers from the control basis, such as the grasping controller, may generate desired end-effector positions in short time intervals. Each desired position requires a global motion generation operation. This poses a significant challenge to existing motion planners, which generally exhibit a high computational complexity for robots with many degrees of freedom. To overcome these limitations, present for even the most efficient global path planners, we have developed the decomposition-based motion planning approach [2]. It is based on the assumption that only a small portion of the free configuration space is relevant to motion planning – large portions of it represent configurations of the robot that are not physically attainable, are obviously not part of a reasonable solution (the robot wrapping around a chair, for example), or are part of a large set of slight variations of relevant configurations. In order to be able to differentiate between relevant and irrelevant configurations, information from the workspace has to be employed.

Decomposition-based motion planning divides the overall motion planning problem into two subproblems. First, a low-dimensional problem is solved in the workspace. Subsequently the resulting solution is employed to determine an answer to the original motion planning problem. The first step can be viewed as identifying those regions of the configuration space that are believed to be relevant to the given problem, thus enabling the second step to employ efficient reactive methods to solve the high-dimensional problem. In preliminary experiments the motion of an eleven degree-of-freedom mechanism operating in a multi-room indoor environment with moving obstacle could be generated several times per second [2]. This significant increase in performance is the result of a conscious trade-off between efficiency and completeness and as a consequence the proposed approach can fail, even if a path exists. In these cases, a more computationally expensive planning method can then be employed to solve the motion planning problem.

3.3 Obtaining Control Basis Elements From Human Interaction

For some tasks it might be difficult to devise an appropriate control basis. In this situation we propose to take advantage of human-robot interaction. A human operator demonstrates a particular behavior and machine learning techniques are used to extract a control policy. This control knowledge can subsequently be used as a new controller in the control basis. This section describes a framework for learning useful controllers that may be otherwise difficult to design.

Our learning framework at the lowest level of the control hierarchy utilizes the combination of supervised learning with an actor-critic architecture for Reinforcement Learning (RL). Almost all RL methods that incorporate human input do so by modifying a "value function" only, and embedded within the value function is an implicit representation of the needed controller. Actor-critic methods, on the other hand, modify separate data structures for the controller (the "actor") and the value function (the "critic"). This obviates the need for a costly search for the best-valued action at each control cycle. Moreover, separate data structures allow the actor to be modified directly by standard supervised learning methods. Essentially, the actor learns to mimic the behavior of its supervisor but adjusts this behavior using its own exploratory actions. See [37] for further details.

As an example, Figure 5 shows a sequence of frames during a simulated peg insertion task. With no initial control knowledge about the task, the actor is completely dependent upon teleoperation input from the human supervisor (via

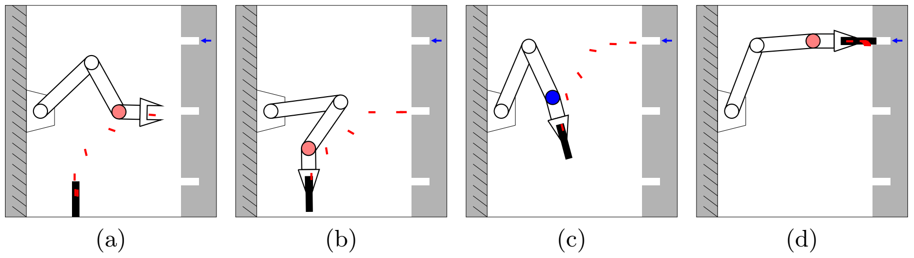

Fig. 5. Screen shots from a simulated peg insertion task. (a) Successful re-grasp of the peg is predicted by the actor. (b) With no knowledge of the target, the actor makes initial progress toward the middle slot. (c) A small correction by the human operator places the robot on track for the upper target, after which (d) the learned controller completes the sub-task.

a mouse). After several trials, however, the actor has gathered sufficient information with which to propose useful actions. Short bars in the figure depict the effects of these actions, as projected forward in time by prediction through a kinematic model. In the leftmost panel, for instance, the bars indicate to the operator that the learning system will translate and rotate the end effector for successful re-grasp of the peg. In this scenario, the actor has no knowledge of the target slot and so immediately after re-grasp (panel b) the learning system "proposes" insertion into the recently visited middle slot. A momentary command from the operator is sufficient to push the system into the basin of attraction for the upper target (panel c). Finally, the rightmost panel shows successful completion of the sub-task a short time later under full control by the actor. This approach has been found to be effective in a simulated peg-in-hole task. We are currently attempting a hardware implementation on the UMass Humanoid.

4 Perceptual Operators

Relevant perceptual information to inform robot decision making is critically important to robot control. In the Control Framework, perceptual information is processed by a hierarchy of perceptual operators that comprise the perceptual basis. These perceptual operators extract continuous information from sensors and in some cases match it against a set of task-relevant features. Continuous data may be feedback for low-level controllers while discrete information can inform DEDS-level decision making.

The approach to sensory processing in the proposed framework is based on the assumption that sensor data can provide expressive and relevant state information. A key question in dealing with the complexity of processing image streams is how to focus one's attention on the aspects of the images that are relevant to performing a particular task. Since it is often impossible to establish a small set of meaningful visual features that are relevant to a variety of tasks in open environments, we feel that it is important to not commit *a priori* to particular, high-level visual operators. Instead, our approach is to employ an adaptive perceptual system that generates its own features by composing primitive features into high-level features that then serve as an input to a controller from the control basis [28, 29].

The primitive features that we use are local, oriented, and appearance-based. Oriented derivatives of 2D Gaussian functions are used to form a steerable basis. We then normalize features for orientation on the image plane to achieve rotational invariance [36]. Primitive features are *texels* – a vector consisting of filter responses from several Gaussian derivative operators at multiple scales. Texels are oriented by steering the responses of the first derivative operators. Spatial combinations of these primitives can express a wide variety of shape and texture characteristics at various degrees of specificity. We compose features on the image plane using geometric, topological, conjunctive, and disjunctive relations between features.

The search for discriminating features is focused on salient regions of the image plane (such as regions containing many edges) and constructs a Bayes net classifier for estimating the conditional probabilities of features with respect to important visual categories or classes [28]. The discriminative power of a composite feature is measured in terms of the Kolmogorov-Smirnoff distance between two conditional distributions of a random variable given the value of this variable under two conditions [27]. This learning approach can also be used to acquire visual features that discriminate important, task dependent categories by looking for distinctive differences in the value of our reaching, grasping, and manipulation policies.

We have also demonstrated in simulation a first step toward the learning of visual features that are informative for a grasp control process [5, 30]. In the experiment, the control system is presented with a variety of objects that are to be grasped (cylinders, cubes, and triangular prisms of various sizes and orientation). An image of the object is stored and the grasp controllers are then engaged. The task of the visual feature learning system is to acquire feature constellations that predict both the configuration of the wrist (as discovered by the grasp controllers) and the quality of resulting grasp.

Figure 6 demonstrates one set of features that was learned for triangular prisms. In this case, two distinct feature constellations were discovered, consisting of two and three primitives, respectively. Once a prediction can be performed robustly, the vision system is in a position to recommend an initial hand and wrist configuration from which the grasp controllers may begin to search for a stable grasp. We anticipate that using this hint from the vision system can substantially reduce this search time. This learned mapping from visual features to parameters for a motor act constitutes an affordance for grasping [10, 9].

The discovery of task-relevant visual features extends the capacity of the whole system. In essence, the new perceptual information creates state in higher-level DEDS models. Policies in these DEDS models are now equipped to handle situations not originally planned for in the design of the system.

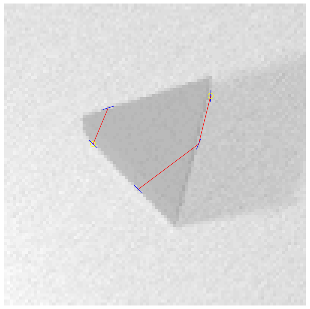

Fig. 6. Grasp-relevant features that were discovered for triangular prisms. The features capture the overall shape of the object, imply the use of a three-fingered grasp, and can be used to determine a good initial hand configuration for the grasp controller.

5 Case Studies of Policy Formation

Controllers for grasping and motion planning provide the necessary components for solving a wide range of reach-and-grasp tasks. Task planning occurs at a discrete events dynamic systems (DEDS) level. In this section, we present examples of this approach applied to a grasping task, a humanoid pick-and-place task, and a tracking and localization task.

5.1 Learning Successful Grasping Behavior

A single grasp controller ofter fares well on its own. However, grasp performance can be improved with a DEDS control policy for activating grasp controllers based on object-specific information. This is possible because grasp controller phase trajectories provide singularly relevant information regarding the grasp process.

When a particular grasp controller is active, it produces a sequence of contact locations, and hence a set of points in the controller phase space, described minimally as the controller error and the change in error. The resulting path through this phase space will often vary dramatically with different object geometries and with different initial conditions of the same object. However, paths representing the same environmental context can be combined to form a model of prototypical system behavior. Through the construction of path models, we recognize object shapes as the grasp controller searches for a stable grasp.

Figure 7 depicts a set of hypothetical models corresponding to policy π_1. If each model is given a discrete label ($M_0 ... M_4$ in the figure), one can describe the transitions between subsets of models in terms of a discrete graph, shown in the right panel of figure 7. The resulting representation defines a discrete state representation that describes the evolution of the grasp controller as a function of object geometry.

In a grasping experiment, we have taken a reinforcement learning approach to acquiring a DEDS-level policy that switch between grasp controller parameterizations (a three-fingered grasp, and three different two-fingered grasps) so as to quickly reach a stable grasp that requires the least amount of friction in order to pick up the object [6, 11]. The total of 61 haptic models were constructed

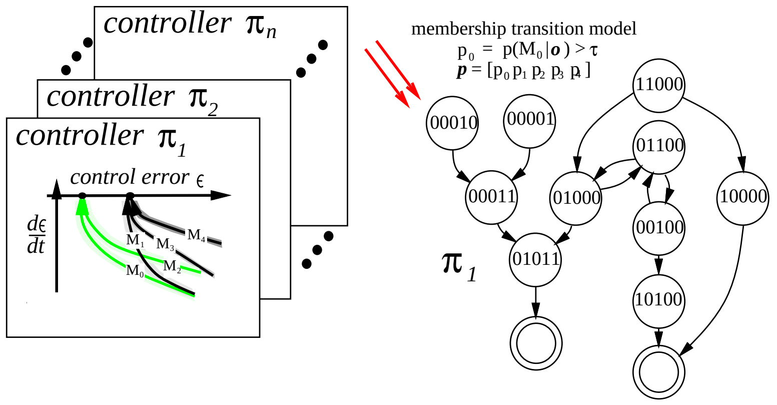

Fig. 7. Each control policy, π_i, yields characteristic phase plane behavior (left). This behavior can be summarized as a set of transitions through a discrete set of path models (right).

by allowing each grasp controller to come to equilibrium from randomly-chosen initial conditions on a set of three object classes (cylinders and rectangular and triangular prisms). Following haptic model construction, Q-learning (a form of reinforcement learning) was employed to learn a grasp controller switching policy based on interacting with the same set of objects. The experimental results demonstrate that a learned policy that depends on haptic state to switch between grasp controllers can achieve higher quality and more consistent grasps over the individual grasp controller parameterizations. The results also suggest an important role for the use of *interaction-based state* in the formation of robust control policies.

5.2 Prospective Pick-and-Place Behavior

The problem of grasping an object and moving it to another location has long been studied in robotics. One approach to this problem is to explicitly compute "pick-and-place" constraints and to perform a search within the constrained space [15, 22]. In this work, it is acknowledged that constraints imposed late in a multi-step control process can influence decisions made early in that process. The classical example is the selection of an initial grasp of a peg that is compatible with a subsequent insertion of that peg into a hole. If the grasp involves surfaces of the peg that must fit into or mate with corresponding surfaces in the hole, a re-grasp must be employed to free those surfaces. The traditional approach uses a backward chaining algorithm that propagates the final assembly process backward in time until this process "finds" the initial state. Such an approach can be computationally expensive and often requires complete models of the task exist prior to acting.

In contrast, humans are capable of robustly planning and executing grasps to objects about which their knowledge is incomplete. Furthermore, it appears that grasping strategies are acquired incrementally as a function of experience with different objects. For example, McCarty *et al.* studied the initial reach made by infants to a spoon laden with applesauce [24]. The youngest infants (9 months) demonstrated an almost “reflexive” strategy in which they grasped the spoon with their dominant hand and immediately brought their hand to their mouth. This strategy is successful when the spoon is presented with the bowl of the spoon on the thumb side of the hand, but fails when the spoon is presented in the opposite orientation. In the latter case, the infants corrected their movement by either regrasping the spoon or rotating their hand into an awkward configuration. With age, the policy evolves to an anticipatory regrasping strategy that predicts which arm to use so that regrasping is not necessary.

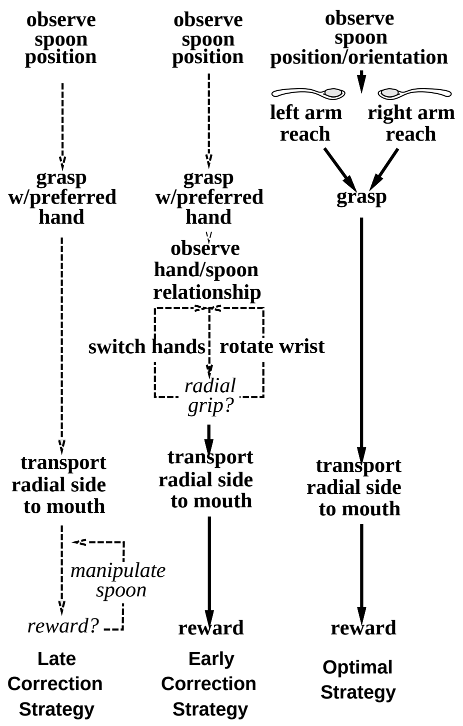

Fig. 8. Prospective behavior in the applesauce experiment. Dotted lines indicate exploration and solid lines indicate a developing policy.

We investigate these findings by implementing a robot anologue of the child study. A robot must grasp an object and insert it into a receptacle. We use a DEDS layer based on closed-loop reach and grasp controllers. Gross motion controllers use harmonic path planning in configuration space [8] to implement large movements of the arms while providing collision avoidance. Fine motion controllers implement Cartesian motions to perform initial positioning of the hands for grasping. Grasp controllers use contact position and normal feedback to minimize force and moment residuals [7], resulting in statically stable grasps. In addition, a visual operator characterizes the object and provides position information to the DEDS layer.

We implemented Q-Learning to select policies in the DEDS framework [39, 41, 12]. Simulation runs were performed for scenarios with and without a “dominant hand”. We gave the agent a dominant hand strategy by pretraining it for 400 trials with the object always presented in the same orientation. After pretraining, the agent was analogous to the youngest children in the child study. It reached for the object with a single hand without regard for the visual input. As with the children, it must learn to suppress the dominant strategy to perform optimally on difficult trials. Next, the agent performed 600 trials with the object presented randomly in one of two orientations.

When experience is limited to a single object orientation, the robot develops a reflexive strategy in which a "dominant" hand is always used to grasp the object. When the object is then presented in either orientation (simulating the commencement of the infant experiment), the robot responds with a mixture of dominant and non-dominant hand strategies. With further experience, strategies requiring correction give way to the most efficient strategy as the robot comes to discover the features that capture the key visual differences between the two conditions.

The "developmental trajectory" exhibited by the model is similar to that observed by McCarty *et al.* At the beginning of the hard trials when the initially learned dominant strategy fails, the agent learns the late correction strategy. Eventually, the robot learns what visual information is relevant to the task and discovers the optimal strategy.

5.3 Hierarchy in DEDS

While policies for prospective pick-and-place behavior can be learned in a single DEDS layer, it is often useful to utilize a hierarchy of policies. For example, pick-and-place control knowledge can be useful in a variety of different manipulation tasks. If the pick-and-place knowledge already exists at one level of the DEDS hierarchy, then it can be used at higher levels as a single, atomic action.

Another type of control knowledge that is useful at high levels in the DEDS hierarchy is perceptual localization and tracking. Many tasks require a robot to visually locate and characterize people, other robots, obstacles, and graspable objects in the environment. As with pick-and-place control knowledge, control policies for tracking and localization can be learned or encoded at low levels in the DEDS hierarchy and used by higher levels.

Perceptual tracking and localization can be difficult because the limited supply of sensor resources and processing time makes deciding which perceptual operators to use a difficult decision making problem. Decisions concerning which perceptual operators to use greatly influences the system's success in locating and tracking significant events. For example, spending a lot of time extracting complex features from a fast moving object could cause the system to lose track of the object. This decision problem can be solved by a policy in the DEDS framework. The tracking policy utilizes perceptual features specifically relevant to tracking and uses this information to control the camera.

DEDS policies for tracking and localization are embedded in Fault Containment Units (CUs). CUs are a software engineering architecture for modularizing robust functionality [4]. In this case, CUs implement DEDS policies for robustly tracking and localizing visual or auditory information. Using CUs clearly specifies the system requirements for each level in the DEDS hierarchy.

CUs responsible for gathering perceptual information operate at different levels in the DEDS hierarchy. At the lowest level, CUs are made up of one or more control policies that coordinate the behavior of a particular set of resources

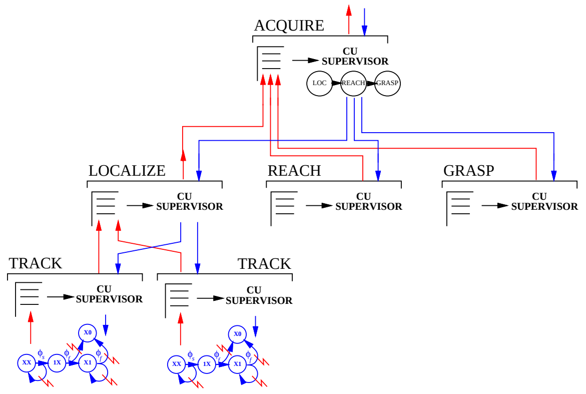

Fig. 9. A DEDS hierarchy implementing localize, reach, and grasp functionality. Arrows pointing up describe the path of information reported by CUs. The CU Supervisor implements the CU DEDS model. The arrows pointing down from CU Supervisors indicates the path of command information.

to service a primitive information request. For example, a tracking CU might saccade to interesting perceptual events. If a sensor fails, the CU selects a new sensor to provide the same type of information and informs the client process of the error. CUs at higher levels accomplish their objectives by using subordinate CUs as resources to do the necessary subtasks. For example, a localize CU might instantiate at least two tracking CUs to form a virtual stereo pair and triangulate on the subject.

Pick-and-place control knowledge can be combined with localization and tracking control policies in a single DEDS model. Figure 5.3 shows one such hierarchy. By combining CUs for localization, reaching, and grasping, the robot is re-using important control knowledge in each of these independent domains. The acquire CU can associate information returned by the localize CU with resource decisions in subsequent reach and grasp tasks. For example, scale feedback from the localize CU can inform a the acquire CU regarding good choices for reach and grasp controllers.

6 Conclusion

Autonomous control of humanoids robots in open environments is an important, but elusive objective. Although machine learning methods applicable to robot control exist, there are few examples of autonomous systems that are not programmed for very specific situations. There are several reasons for this. First, in unstructured domains, it is difficult to specify task objectives at an abstract level or re-use control knowledge. Another problem is the fundamental multi-objective nature of robot control. For example, while attempting to grasp an object with a robot hand, it is often necessary for the arm, hand, and fingers to avoid obstacles. In addition, it is often difficult for the system to acquire the perceptual information necessary to adequately solve a task. Visual occlusions or noise may render hard-coded perceptual features useless.

The hierarchical DEDS approach accrues the many benefits of abstraction. Hierarchy allows a complex system function to be described in terms of simple units of control. This can dramatically decrease the learning time for machine learning methods such as reinforcement learning. Task level policies are easily described in terms of re-usable controllers and DEDS control policies.

Closed-loop controllers are well suited to robust multi-objective control. For example, the task of grasping while avoiding obstacles cited above can be concisely described by projecting the grasp control law into the nullspace of a collision avoiding control law.

Our framework provides two mechanisms for acquiring task-relevant information for high-level control policies. First, we show how complex visual features can be discovered based on their ability to predict successful task outcomes. Second, controller trajectory can provide uniquely relevant information regarding the state of the overall system. These features grow the state representation at the decision-making level and expand the learning capability of the system.

References

1. Robert Bohlin and Lydia E. Kavraki. Path planning using lazy PRM. In *Proceedings of the International Conference on Robotics and Automation*, volume 1, pages 521–528, San Francisco, USA, 2000.
2. Oliver Brock and Lydia E. Kavraki. Decomposition-based motion planning: A framework for real-time motion planning in high-dimensional configuration spaces. In *Proceedings of the International Conference on Robotics and Automation*, pages 1469–1474, Seoul, Korea, 2001.
3. Oliver Brock and Oussama Khatib. Elastic strips: A framework for motion generation in human environments. *International Journal of Robotics Research*, 21(12):1031–1052, 2002.
4. Jamieson M. Cobleigh, Leon J. Osterweil, Alexander Wise, and Barbara Staudt Lerner. Containment units: A hierarchically composable architecture for adaptive systems. In *Proceedings of the 10th International Symposium on the Foundations of Software Engineering*, pages 159–165, 2002.
5. J. Coelho, J.H. Piater, and R.A. Grupen. Developing haptic and visual perceptual categories for reaching and grasping with a humanoid robot. In *First IEEE-RAS International Conference on Humanoid Robots*. IEEE, September 2000.
6. J. A., Jr. Coelho and R. A. Grupen. Learning in non-stationary conditions: A control theoretic approach. In *Proceedings of the Seventeenth International Conference on Machine Learning*, Stanford, CA, 2000.
7. Jefferson A. Coelho Jr. and Roderic A. Grupen. A control basis for learning multifingered grasps. *Journal of Robotic Systems*, 14(7):545–557, 1997.
8. C. Connolly and R. Grupen. On the applications of harmonic functions to robotics. *Journal of Robotics Systems*, 10(7):931–946, 1993.
9. A. H. Fagg. *A Computational Model of The Cortical Mechanisms Involved in Primate Grasping*. PhD thesis, Department of Computer Science, University of Southern California, 1996.
10. A. H. Fagg and M. A. Arbib. Modeling parietal-premotor interactions in primate control of grasping. *Neural Networks*, 11(7/8):1277 – 1303, 1998.
11. R. Grupen and J. Coelho. Acquiring state from control dynamics to learn grasping policies for robot hands. *International Journal on Advanced Robotics*, 16(5):427–444, 2002.

12. M. Huber and R.A. Grupen. A hybrid discrete event dynamic systems approach to robot control. Technical Report 96-43, University of Massachusetts Amherst Computer Science Department, October 1996.
13. Manfred Huber and Roderic A. Grupen. A feedback control structure for on-line learning tasks. *Robots and Autonomous Systems*, 22(3-4):303–315, 1997.
14. John Jameson and Larry Leifer. Automatic grasping: An optimization approach. *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-17(5), September 1987.
15. J.L. Jones and T. Lozano-Pérez. Planning two-fingered grasps for pick-and-place operations on polyhedra. In *Proceedings of 1990 Conference on Robotics and Automation*, pages 683–688. IEEE, May 1990.
16. Lydia E. Kavraki, Peter Švestka, Jean-Claude Latombe, and Mark H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.
17. Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98, 1986.
18. Oussama Khatib. A unified approach to motion and force control of robot manipulators: The operational space formulation. *International Journal of Robotics and Automation*, 3(1):43–53, 1987.
19. Daniel Koditschek and Elon Rimon. Robot navigation functions on manifolds with boundary. *Advances in Applied Mathematics*, pages 412–442, 1990.
20. John E. Laird, Paul S. Rosenbloom, and Allen Newell. Chunking in Soar: The anatomy of a general learning mechanism. *Machine Learning*, 1:11–46, 1986.
21. Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, 1991.
22. T. Lozano-Pérez. Automatic planning of manipulator transfer movements. *IEEE Transactions Systems, Man, and Cybernetics*, 11(10):681–689, 1981.
23. S. Mahadevan and J. Connell. Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, 55:311–365, 1992.
24. M. E. McCarty, R. K. Clifton, and R. R. Collard. Problem solving in infancy: The emergence of an action plan. *Developmental Psychology*, 35(4):1091–1101, 1999.
25. A. Newell and H.A. Simon. *Human Problem Solving*. Prentice Hall, Englewood Cliffs, NJ, 1972.
26. C.M. Özveren and A.S. Willsky. Observability of discrete event dynamic systems. *IEEE Transactions on Automatic Control*, 35(7):797–806, 1990.
27. J.H. Piater and R.A. Grupen. Toward learning visual discrimination strategies. In *Proceedings of the IEEE Computer Society on Computer Vision and Pattern Recognition*, pages 410–415. IEEE, June 1999.
28. J.H. Piater and R.A. Grupen. Feature learning for recognition with bayesian networks. In *Proceedings of the Fifteenth International Conference on Pattern Recognition*. IEEE, September 2000.
29. Justus Piater. *Visual Feature Learning*. PhD thesis, University of Massachusetts, 2001.
30. Justus H. Piater. Learning visual features to recommend grasp configurations. Computer Science Technical Report 2000-40, University of Massachusetts, Amherst, MA, July 2000.
31. R. Platt, A. Fagg, and R. Grupen. Extending fingertip grasping to whole body grasping. In *Proceedings of ICRA'03*, 2003.
32. 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.

33. J. Ponce, S. Sullivan, A. Sudsang, J. Boissonnat, and J. Merlet. On computing four-finger equilibrium and force-closure grasps of polyhedral objects. *Int. J. Rob. Res.*, 16(1):11–35, 1996.
34. Doina Precup and Richard S. Sutton. Multi-time models for temporally abstract planning. In *Proceedings of Advances in Neural Information Processing Systems 10*. MIT Press, 1997.
35. Peter J.G. Ramadge and W. Murray Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–97, January 1989.
36. Rajesh P. N. Rao and Dana H. Ballard. An active vision architecture based on iconic representations. *Journal of Artificial Intelligence Research*, 78:461–505, 1995.
37. Michael T. Rosenstein and Andrew G. Barto. Supervised learning combined with an actor-critic architecture. Technical Report 02–41, Department of Computer Science, University of Massachusetts, 2002.
38. M. Sobh, J.C. Owen, K.P. Valvanis, and D. Gracani. A subject-indexed bibliography of discrete event dynamic systems. *IEEE Robotics & Automation Magazine*, 1(2):14–20, 1994.
39. R. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps. *Artificial Intelligence*, 112:181–211, 1999.
40. Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and Semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211, 1999.
41. R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.
42. S. Uppala, D. Karuppiah, M. Brewer, S. Chandu Ravela, and R. A. Grupen. On viewpoint control. In *IEEE Int'l Conference on Robotics and Automation*, May 2002.