Decoupling Behavior, Perception, and Control for Autonomous Learning of Affordances

Tucker Hermans James M. Rehg Aaron F. Bobick

***Abstract*— A novel behavior representation is introduced that permits a robot to systematically explore the best methods by which to successfully execute an affordance-based behavior for a particular object. The approach decomposes affordance-based behaviors into three components. We first define *controllers* that specify how to achieve a desired change in object state through changes in the agent's state. For each controller we develop at least one *behavior primitive* that determines how the controller outputs translate to specific movements of the agent. Additionally we provide multiple *perceptual proxies* that define the representation of the object that is to be computed as input to the controller during execution. A variety of proxies may be selected for a given controller and a given proxy may provide input for more than one controller. When developing an appropriate affordance-based behavior strategy for a given object, the robot can systematically vary these elements as well as note the impact of additional task variables such as location in the workspace. We demonstrate the approach using a PR2 robot that explores different combinations of controller, behavior primitive, and proxy to perform a push or pull positioning behavior on a selection of household objects, learning which methods best work for each object.**

I. Introduction

As the goal of having robots operate in uncontrolled environments becomes more critical to the advancement of robotics, there has been much research on the notion of *affordances* of objects with respect to a robot agent [1]. Within the context of robotics, affordances describe the possible actions an agent can take acting upon an object and the resulting outcome [2]. Specific examples might include *graspable* (e.g. [3]) or *pushable* [4] that indicate a particular object can be grasped or pushed, respectively. Because one can cast affordances as state-action pairs that will transform the object state in some way, there has been further work in considering affordance as a basis for planning [5]. If the robot has a goal of clearing the path to an object being fetched, it might first push interfering objects to the side assuming they can be pushed, i.e. have the affordance *pushable*.

However, while a planner may be able to leverage an abstracted description of the affordance as being true or not of an object, such a high level description is not sufficient to actually *execute* the action required for the affordance. And, indeed the method of performing the action may vary by object or object state: pushing a round cereal bowl might be quite different than pushing a TV remote control that has rubber buttons that occasionally stick to a table surface.

Tucker Hermans, James M. Rehg, and Aaron F. Bobick are affiliated with the Center for Robotics and Intelligent Machines and The School of Interactive Computing, Georgia Institute of Technology, Atlanta, GA. {thermans, rehg, afb}@cc.gatech.edu This work was supported in part by NSF Award 0916687.

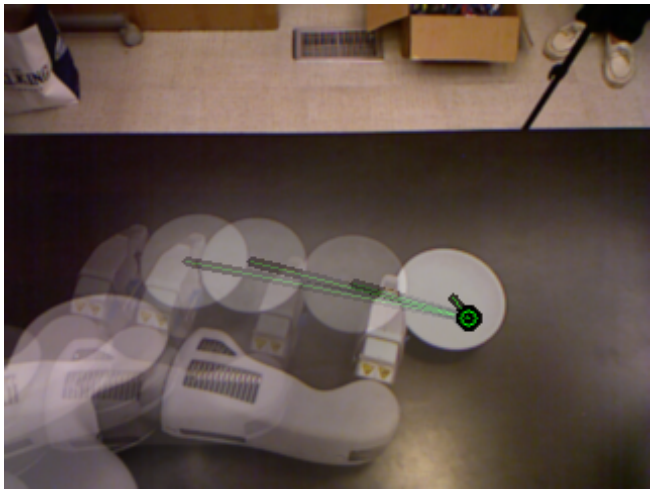

Fig. 1: Composition of example frames of the robot performing a successful pushing behavior. The green line denotes the vector from the estimated centroid of the object to the goal location.

The goal of this paper is to introduce a mechanism by which a robot determines whether a given object has a particular affordance by systematically exploring *how* the robot should behave to successfully achieve the goal defined by the affordance. We do this by decomposing an affordance-based behavior into three components. We first define *controllers* (control policies) that specify how to achieve a desired change in object state through changes in the agent's state. For each controller we develop at least one *behavior primitive* that determines what configuration the robot maintains of those degrees of freedom not directly accounted for by the controller. In Figure 1 the behavior primitive has the robot keep the palm of the hand orthogonal to the table, while the controller commands the x and y velocities of the hand. Additionally we provide multiple *perceptual proxies* that define the representation of the object which will be used to compute the control signal. The proxy must be sufficiently rich to support estimation of the variables required by the controller. One novelty of our approach is that multiple proxies may support the same control policy and a given proxy representation may be selected for use with more than one controller. Additionally, a single behavior primitive may be compatible with multiple controllers. Decoupling these components allows the systematic exploration of a variety of strategies when evaluating the affordances of novel objects. The composition of these components allows for a large variability in robot behavior from a relatively small amount of programming.

In this paper we use as examples the affordances of *push-positionable* and *pull-positionable* where the goal is to move the target object to a specified location on a table. We develop three different feedback controllers to implement these actions. Each of the two push-positioning controllers admits any of three behavior primitives of *overhead push*, *fingertip push*, and *gripper sweep*. Additionally each controller can choose to use one of several perceptual proxies. These methods require no prior knowledge of the object being pushed and make no estimates of underlying model parameters. We show how a robot can autonomously determine the effectiveness of a particular affordance-based behavior combination of proxy, controller, and behavior primitive for a given novel object. We examine which methods perform best for fifteen different household objects and explore the success and failure of the approaches as a function of where in the robot's workspace the object is located. Figure 1 shows the robot successfully push positioning a dinner bowl using the affordance-based behavior combination of centroid perceptual proxy, centroid alignment controller, and gripper sweep behavior primitive.

We organize the remainder of our paper as follows. Section II describes relevant past work on the topic of affordance learning and affordance-based planning; we also briefly mention prior methods of robotic control for pushing. In Section III we formally define our use of affordances and the push and pull positioning tasks. Section IV presents our three proposed feedback controllers whose effectiveness, as we will see, varies depending upon object. We give details of our implemented perceptual proxies in Section V followed by the implemented behavior primitives in Section VI. Section VII presents results of over 1500 affordance-based behavior trials performed semi-autonomously by a robot using our proposed system. We conclude with directions for future work in Section VIII.

II. Related Work

We divide prior work into efforts related to the general problem of affordance learning and work about developing the specific affordance of pushing.

Affordance Learning

In early work on affordance prediction described in [6, 7], a humanoid robot learns to segment objects through actions such as poking and prodding. After interaction with a set of objects, the system could learn the rollable affordance for the objects and predict the result of hand-object interactions. The goal was to learn parameters such as initial location of the hand with respect to the orientation of an object that best induce the desired motion. The actions were atomic in the sense that they were applied in their entirety and the results measured. In [8], a classification method is applied to high-level image features to learn the affordance of liftable. Using decision tree classifiers with SIFT and patch features, they demonstrate the ability to learn liftable vs nonliftable objects. Sun et al. use object category prediction to aid in predicting affordances, while Hermans et al. use object attributes as a predictive layer between image features and affordances [9, 10].

A series of works [11–13] addresses the task of recognizing the graspable and tappable affordances, based upon experimentation through self-observation of actions. Learning in a Bayesian network is employed to learn cuing rules for actions. The network models the relationship between object appearance and motion, end-effector motion, and action. In [14], a functional approach to affordance learning is developed in which subcategories of the graspable affordance (such as handle-graspable and sidewall-graspable) are learned by observation of human-object interactions. Ugur et al. also examine grasping as an affordance prediction problem, by first creating affordance labels through unsupervised clustering of grasping attempts [15]. A support vector machine is then trained to predict the affordance label as a function of the object and a parameter of grasping behavior used. Krömer et al. predict the probability of grasps being successful at specific locations on an object using visual kernel methods [16].

With respect to planning, affordance-based modeling of robot-object interaction would allow a planning system to systematically select from a set of actions to achieve desired sub-goals. An example of such an approach is given in [5] where the robot arranges plates and bowls on a table. However, there is an assumption of a priori knowledge about which behaviors can successfully operate on which objects and what the resulting state of the action will be.

The concept of Instantiated State Transition Fragment (ISTF) is introduced in [17]. It encodes the pairing between an object and an action in the context of the state transition function for a domain-specific planner. The authors describe a process of learning Object Action Complexes (OACs) through generalization over ISTFs. Montesano et. al. [13] use learned OACs in planning and executing a multi-step task. Ugur et al. present a planning architecture where behaviors define the actions of the planner and state transition outcomes are the learned outcomes of object affordances [18]. Full plans to achieve higher level tasks, such as lifting an object, are pieced together through chaining of affordance predictions. Finally, while not explicitly mentioning affordances, Klank et al. examine how a robot can choose from different perceptual and manipulation mechanisms to more reliably achieve a task on different objects in different scenarios [19]. That approach is related to the method of perceptual proxies developed here.

Pushing Behaviors

Effective pushing behaviors offer a number of benefits in robotics domains which complement standard pick-and-place operations. For example pushing can be used to move objects too large for the robot to grasp, to more quickly move objects to new locations, or to move an object while another object is already grasped. As such there has been considerable work at developing such capability. Early work that analyzed a complete model of the dynamics of pushing was developed by Mason who describes the qualitative rotational changes of sliding rigid objects being pushed by either a single point or single line contact [20]; representative examples of some

more recent applications of pushing are available in [4, 21–26].

Notably, Ruiz-Ugalde et al. execute a pushing behavior by determining the static and kinetic friction coefficients for multiple objects with rectangular footprints, both between the robot hand and object and between the object and table [26]. Additionally they present a robust controller using a cart model for the object being pushed. Their control is the closest approach we have found to the pushing controllers presented in this work. However, their overall approach presumes the ability to predict the resulting action based upon known or learned parameters that characterize the physics of the object.

To address the inherent difficulty in estimating model parameters, there are data-driven methods that use an empirically derived characterization of the outcomes of specific actions applied to the object. For example, Narasimhan uses vision to determine the pose of polygonal objects of known shape in the plane [27] and then develops a variety of methods to push objects into the desired location and orientation including a data driven approach that learns the effect of different pushes on the object. Similarly, Salganicoff et al. present a method for learning and controlling the position in image space of a planar object pushed with a single point contact [28]. Slip of the object is avoided by pushing at a notch in the object. Scholz and Stilman learn object specific dynamics models for a set of object through experience [29]. Each object is pushed at a number of predefined points on the perimeter and the robot learns Gaussian models of displacement of the object's 2D pose, (x, y, θ), at each location. These learned models are then used to select the input push location given a desired object pose.

III. Problem Statement

We define an affordance to exist between a robot and an object, if the robot can select a specific behavior primitive, controller, and perceptual proxy by which it can successfully perform the desired action. We take as example actions those of *push positioning* and *pull positioning*, where the robot must position an object at an arbitrary location by pushing or pulling with its arm. We assume that the object is being manipulated over a plane and thus the object state $X = (x, y)$ defines the location of the origin of the object in a 2D space. We denote the goal pose as $X^* = (x^*, y^*)$. This state representation is sufficient for a task level planner, however, a specific controller may require more state variables to be estimated by the relevant perceptual proxy.

The (unknown) dynamics of the pushing system are governed by the nonlinear relation $\dot{X} = h(X, Q, U)$ which defines the interaction dynamics between the object state, the robot configuration Q, and the input to the robot U. Importantly, we make no attempt to model h. In developing our visual feedback controllers, we presume we do not have an exact measurement of the object state. Instead we will operate on the estimated state $\hat{X}$ that will be computed at each timestep based upon properties of a perceptual proxy. In this work we control the arm through Cartesian control, both position and velocity, in the robot's task frame. We denote the specific forms of U and X used in our controllers in detail below. Our task thus becomes defining a feedback control law $U = g(\hat{X}, X^*)$ which drives the position error $X_{err} = X^* - \hat{X}$ to zero.

IV. Feedback Controllers: Pushing and Pulling

In this section we define several visual feedback controllers for pushing and pulling. Each controller we develop defines a necessary set of state variables. This state will be estimated at each time step using an appropriate perceptual proxy described below in V. The control outputs are used to command an appropriately selected behavior primitive described in VI. More details of the pushing controllers can be found in [30].

A. Spin-Correction Control

Our first method of defining a push-positioning controller relies on the fact that the direction of an object's rotation while being pushed depends on which side of the center of rotation the applied force intersects. This fact is well described by the limit surface formulation [20, 31]. Mason derived the velocity direction of a sliding object as a function of the forces applied by the pushing robot as well as the support locations and mass distribution of the object [20]. These parameters are difficult to know or estimate well for a given object and even when they are known, the exact behavior is often indeterminate [20]. However, we make use of Mason's observation that the rotation of the object abruptly changes direction when the input force passes directly through the center of rotation of the object. As such we can use the direction of the observed rotation of the object to infer which side of the center of rotation the applied forces are currently acting through. We can then correct the direction of our applied forces to compensate for any unwanted rotation of the object.

To reduce induced rotation, our controller attempts to push the object through its center in the direction of the goal position. This gives a simple procedure for determining the initial hand position. We cast a ray from the goal location through the centroid of the object and find its intersection with the far side of the object. This location defines the initial position for the hand. We further orient the hand so that the desired portion of the gripper is facing in the direction of the goal from the initial position (cf. Section VI). An example image of the initial hand placement can be seen in the upper left of Figure 2. Once positioned our feedback control process is initiated. The controller is defined in equations 1 and 2 which operates on state $X = (x, y, \theta, \dot{\theta})$ and computes input U of x and y velocity of the end effector in the robot's workspace.

$$u_{\dot{x}} = k_g e_{goal_x} - sin(\phi_g)(e_{rot}) \quad (1)$$

$$u_{\dot{y}} = k_g e_{goal_y} + cos(\phi_g)(e_{rot}) \quad (2)$$

Our control is comprised of two terms. The first pushes through the object driving it to the desired goal, while the second displaces the contact location between the robot and object to compensate for changes in object orientation. The input control defined in equation 3 commands the robot to push in the direction of the goal. The overall effect of this

component is controlled by the positive gain k_g. Since the object lies between the end effector and the goal this causes the object to translate towards the goal.

$$e_{goal_x} = (x^* - \hat{x}),\ e_{goal_y} = (y^* - \hat{y}) \tag{3}$$

However, since the forces applied by the robot on the object are not pushing directly through the center of rotation, the object will undoubtedly spin. To compensate for this we apply additional input velocities proportional to the observed rotational velocity of the object. We desire not only that the object not rotate, but also that it maintains its initial orientation θ_0. We combine these terms to generate e_{rot}.

$$e_{rot} = k_{sd}\dot{\hat{\theta}} - k_{sp}(\theta_0 - \hat{\theta}). \tag{4}$$

We desire to displace the end effector perpendicular to the current direction of the object's translational motion. Since our estimate of the instantaneous velocity is somewhat noisy, we instead rotate the velocity vector about the angle defined between the center of the object and the goal ϕ_g.

$$\phi_g = \texttt{atan2}(y^* - \hat{y}, x^* - \hat{x}) \tag{5}$$

Our pushing controllers halt once $x_{err} < \epsilon_x$ and $y_{err} < \epsilon_y$. For the purpose of developing this method as well as the controller in Section IV-B, the gains are manually adjusted, but remain fixed for all objects.

B. Centroid Alignment Control

Our second push-positioning controller replaces the monitoring of object orientation with a strategy based upon the relative locations of the object's centroid, the assumed location of the contact point on the end effector, and the goal position. This allows for control of objects where estimating a dominant orientation is difficult, such as rotationally symmetric objects.

The robot achieves this behavior by using a control law that includes a velocity term to move toward the goal and a second term that moves the end effector towards the vector defined from the goal location through the object's centroid:

$$u_{\dot{x}} = k_{gc}e_{goal_x} + k_c e_{centroid_x} \tag{6}$$

$$u_{\dot{y}} = k_{gc}e_{goal_y} + k_c e_{centroid_y} \tag{7}$$

where e_{goal_x} and e_{goal_y} are as before. The second term provides the additional velocity term toward the goal-centroid line; $e_{centroid_x}$ and $e_{centroid_y}$ are components of perpendicular vector from the presumed end effector contact point to the goal-centroid line. The robot then pushes in the direction of the goal attempting to maintain this collinearity relation. This controller has the state $X = (x, y)$ and computes the same U as in Section IV-A. Additionally, the end effector is initially positioned relative to the object as above.

C. Direct Goal Pull Control

We implemented a single feedback control law to be used with pulling objects. The controller assumes the object is already grasped by the gripper and simply moves with a velocity proportional to the direction of the goal from the current object centroid. The gripper is placed following a similar procedure to that used in pushing. However, the initial hand placement is chosen to be at the closest location on the object to the goal position. The gripper is opened prior to moving to this initial pose. The gripper then moves forward to surround the object and closes to grasp it. Upon halting of the controller, the gripper opens to release the object and moves backwards to clear the object prior to returning to the ready position.

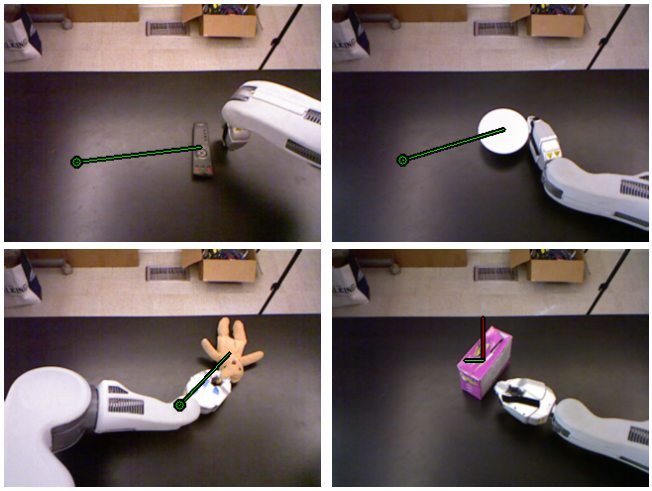

Fig. 2: The first image shows the overhead push behavior primitive placed prior to pushing the television remote. The second image show the sweep push behavior primitive pushing the dinner bowl. The lower left image shows the teddy bear being pulled. The pink box is being pushed with the fingertip push in the final image. The first three images have drawn in green the vector from the current centroid estimate to goal location. The box has the estimated centroid location overlayed.

D. Controller Monitoring

During execution of each feedback controller we monitor for certain conditions where execution should be aborted. The simplest of these is when no object has been detected. To avoid being stuck in strange configurations, we abort execution when neither the arm nor the object has moved after a short period of time. Execution stops when the robots gripper moves farther than some predefined threshold from the object in order to stop the robot from continuing control after losing contact with the object. Finally, the robot halts execution for both push controllers when the estimated object centroid is not between the gripper and goal locations.

V. Perceptual Proxies

The above defined controllers have modest perceptual requirements. The orientation-velocity controller requires both the location of the object and its orientation whereas the centroid-driven push controller and pulling controller require only position as defined by the object's centroid. Here we describe concretely our proxies for estimating these properties for a given object.

We begin with a simple depth-based segmentation and tracking method that currently assumes only a single object resting on the sliding surface (a table) is in the scene. The input is the RGB-D image of a Microsoft Kinect though in this implementation only the depth channel is used. We

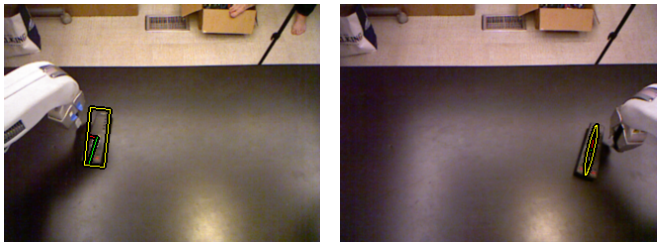

Fig. 3: Examples of the robot performing pushing using feedback from the tracking. The left image shows the state estimated with the bounding box perceptual proxy. The right image displays the estimated ellipse.

initialize the tracker by moving the robot's arms out of the view of the camera, capture the depth image and then use RANSAC [32] to find the dominant plane in the scene parallel to the ground plane. We then remove all points below the estimated table plane and cluster the remaining points. We filter out clusters with very few points and, because we're assuming only one object is on the table, we accept the cluster with most points as the object.[1]

Once initialized we track the object by performing the same procedure with the added step of removing points belonging to the robot from the scene. We project the robot model into the image frame using the forward kinematics of the robot and remove points from the point cloud coincident with the robot arm mask. Because of noise in measurements and other calibration issues points belonging to the robot can sometimes remain. To prevent the tracker from selecting any of these points as the current object we perform nearest neighbor matching between current cluster centroids and the previous object state, selecting the closest as the current object. We then estimate the object velocity using the previous estimate of the object state.

Computing the perceptual proxies needed for each of the controllers is straightforward given the tracker described above. For the centroid based control methods, centroid alignment and gripper pull, we can immediately return the x and y values computed from the centroid of the object point cloud as input. We name this the centroid proxy. However, this estimate may not be the most accurate representation of the objects actual centroid, do to occluded regions of the object and non-uniform mass distributions. As such we implemented two additional proxies to estimate the centroid of the object. The first alternative approach uses RANSAC to fit a sphere model to the object point cloud. We then use the x and y estimates of the sphere's center as input to the controller. This allows for partially occluded spherical objects to have a more stable estimate of the object's center. Our other estimate approach fits the minimum area bounding box to the 2D footprint of the object. We then use the center of this bounding box as the object centroid. While this proxy is not robust to occlusion by the robot, it produces a result that is just a function of the convex hull of the point cloud and is not influenced by how many points occur in any single region of an object's interior.

For the orientation-velocity control we need a proxy that includes an estimate of object orientation, as well as its rotational velocity, with respect to the global robot frame. The bounding box proxy can be used to give us an estimate of the orientation. We simply set the orientation of the object to be the dominant axis of the bounding box. As an alternative proxy we fit a 2D ellipse to the x and y values of all points in the object point cloud and use the orientation of the major axis of the ellipse as the objects orientation θ. In both cases, the change in θ from one frame to the next is the estimated orientation velocity $\dot{\theta}$. Example of computed bounding box and ellipse proxies are shown in Figure 3.

VI. Pushing and Pulling Behavior Primitives

We performed pushing with three behavior primitives: an overhead push, a sweep push, and a fingertip push. These behavior primitives define how the two outputs from the pushing controller, velocity in x and y, are transformed to control the six degrees of freedom of the robot's end-effector. The overhead push has the robot place its hand such that the fingertips are in contact with the table with the wrist directly above. The sweep push places the length of the hand on the table with the flat of the hand facing the object. The fingertip push keeps the palm of the hand parallel to the table and pushes with the tip of the gripper pointing in the push direction. As our controllers operate only within the 2D position of the hand (x, y), the configuration of the end effector with respect to the arm and current table location remain fixed during operation. Specifically this means that the wrist remains above the hand throughout pushing for the overhead push. Likewise the sweep push keeps the long side of the robot hand along the table with the broad side of the hand perpendicular to the surface during pushing and the fingertip push keeps the palm parallel to the table. Images of the robot operating with these behavior primitives can be seen in Figure 2.

For all primitives the arm is moved to the initial pushing pose using Cartesian position control. The arm is first moved to a position directly above the table at the desired pose and desired orientation. The hand is then lowered in a straight line to the initial pushing pose. We use a Jacobian inverse controller to control the Cartesian velocity of the end effector during feedback control.

The gripper pull behavior primitive is similar to the fingertip push primitive, except that the gripper is first opened prior to moving to the initial pose. The gripper then moves forward to surround the object and closes to grasp it. Additionally the initial hand placement is determined to be between the object and goal not behind the object, as in pushing.

We experimentally validate our approach by having a mobile manipulator explore the possible combinations of affordance-based behavior actions over a set of 15 household objects displayed in Figure 4. For each object, the robot attempted at least one left arm and one right arm push or pull action with every possible combination of proxy, behavior primitive, and controller. This produces a set of thirty-two possible affordance-based behaviors.

[1]We note that we [25] and others (e.g. [33]) have previously developed methods for singulating objects from each other by pushing actions, which could be used to perform the necessary object segmentation in clutter.

Fig. 4: The fifteen objects on which the robot performed experiments.

VII. Experimental Validation

For a given object we place it at an initial random position on the table. The robot generates a random goal position at least 0.2 meters from the current location and attempts the first instantiated affordance-based behavior. If the execution succeeds the robot generates a new goal pose as before and attempts the next behavior instantiation with the object positioned at its current location. If the controller aborts execution prior to reaching the goal the robot will reattempt the current combination up to two additional times. If the goal is not reached after the third attempt the robot moves on to the next affordance-based behavior combination. If the object is knocked off the table or out of a predefined workspace for the robot the robot asks a human operator to replace the object and continues exploration. If upright objects were knocked over during the exploration the robot continues to attempt the current affordance-based behavior. However once the robot completes the current trial, either by being successful or by aborting three times, the human operator pauses the search and returns the object to its canonical pose.

We implemented our system on a Willow Garage PR2 robot with a Microsoft Kinect mounted on its head for visual input. In all experiments $\epsilon_x = \epsilon_y = 0.01$ meters. Below we show detailed results of the affordance-based behavior exploration consisting of more than 1500 total push/pull trials.

A. Object Affordances

For any given object we would like to know the best affordance-based behavior to use to effectively move it when a given task demands. Here we define the best affordance-based behavior to be the choice of controller, perceptual proxy, and behavior primitive which together produce the smallest average final position error averaging over trial and workspace location. The best choice of affordance-based behavior combination for each object, along with its statistics, is presented in Table I. We note several results: First, we see that the overhead push with the centroid controller performs the best on average for most objects. Only the telephone and soap box, both of which tend to rotate when pushed, found better average performance with the spin compensation controller. Additionally the chosen behavior primitives were all either overhead push or gripper sweep for these objects. This is explained by the fact that the fingertip push and gripper pull operate well only in restricted areas and angles of the workspace do to the constraint on the hand pose. We see that there is a nearly even split between the use of bounding box and centroid as perceptual proxy. The sphere is a much more specialized proxy that only works well on a few objects with mostly spherical shape.

We can gain further insight into the behaviors by examining which affordance-based behavior combinations produced a final goal error below a specified threshold. The results for all objects are presented in Table II. Here we can examine individual attempts rather than average performance. For example, the TV remote — with it's rubber buttons that grip the surface — could only be controlled by the overhead push behavior using the spin-compensated controller employing the ellipse proxy. Likewise, the shampoo was quite poorly controlled: only one behavior combination (overhead-push, bounding-box, spin-compensation) ever achieved the goal and then only once out of six attempts. Conversely, the pink box could be successfully manipulated using a variety of combinations with only a slight preference for the same control combination as the TV remote. By such detailed experimentation and analysis the robot can develop a set of strategies not only for an initially available set of objects but also, potentially, for novel objects once the robot gains experience with them. For example, once a new object is observed to behave like a Mug with respect to several behavior combinations a robot could rapidly develop a strategy for that new object based upon its familiarity with the Mug's behavior.

B. Affordance-Based Behavior Performance as Function of Workspace

The above analysis averages performance independent of target location under the assumption that the robot can perform these actions equally well throughout its workspace. Of course, mechanical limitations make certain actions more difficult at different positions. For example, the fingertip push behavior primitive has difficulty in pointing the fingertip towards the robot's torso. Additionally it may be impossible for the robot to reach objects far to the right with the left arm. Knowing which operations perform best in different areas of the workspace is difficult to predict. We would prefer learned models of where to apply certain behavior primitives as opposed to the heuristics previously used (c.f. [25]). Having knowledge of which behavior primitive works best in a specific region of the workspace could be helpful in attempting to manipulate a previously unseen object. Behavior primitives that have been seen to consistently fail may be skipped in favor of those more likely to succeed.

To compare the various behaviors, we grouped push trials by their starting (x, y) locations as well as the pushing angle, the angle from the initial object location pointing towards the goal position. We quantized initial x and y locations to the closest 0.05 meter value and angles to the closest of eight directions at $45°$ increments. We visualize the best

Object Name	Primitive	Proxy	Controller	Mean Score	Variance
Plate	Gripper Sweep	Centroid	Centroid Controller	0.085	1.99e-05
Towel	Overhead Push	Centroid	Centroid Controller	0.036	5.40e-05
Hair Brush	Overhead Push	Centroid	Centroid Controller	0.136	0.001
Toothpaste	Overhead Push	Bounding Box	Centroid Controller	0.126	0.015
Pink Box	Gripper Sweep	Centroid	Centroid Controller	0.056	0.002
Shampoo	Overhead Push	Bounding Box	Centroid Controller	0.159	0.032
Telephone	Overhead Push	Bounding Box	Spin Compensation	0.032	5.21e-05
Soap Box	Overhead Push	Bounding Box	Spin Compensation	0.086	0.003
Mug	Overhead Push	Centroid	Centroid Controller	0.034	2.69e-04
Medicine Bottle	Overhead Push	Bounding Box	Centroid Controller	0.032	9.12e-05
Teddy Bear	Overhead Push	Bounding Box	Centroid Controller	0.083	2.15e-04
Red Bottle	Overhead Push	Bounding Box	Centroid Controller	0.084	0.001
TV Remote	Overhead Push	Bounding Box	Centroid Controller	0.147	0.007
Bowl	Gripper Sweep	Centroid	Centroid Controller	0.020	3.52e-05
Salt	Overhead Push	Centroid	Centroid Controller	0.015	1.02e-04

TABLE I: Lowest average final error affordance-based behaviors for the fifteen objects.

Object Name	Primitive	Proxy	Controller	# Within 0.02m	# Within 0.04m	# Attempts
Plate	Overhead Push	Sphere	Centroid Controller	0	2	5
	Overhead Push	Centroid	Centroid Controller	0	1	6
	Fingertip Push	Centroid	Centroid Controller	0	1	4
	Fingertip Push	Sphere	Centroid Controller	0	1	6
Towel	Overhead Push	Bounding Box	Centroid Controller	1	1	5
	Overhead Push	Centroid	Centroid Controller	0	4	5
	Gripper Sweep	Ellipse	Spin Compensation	0	1	6
	Overhead Push	Sphere	Centroid Controller	0	1	6
Hair Brush	Gripper Sweep	Centroid	Centroid Controller	2	3	6
Toothpaste	Overhead Push	Bounding Box	Centroid Controller	2	2	6
	Overhead Push	Ellipse	Spin Compensation	1	1	4
	Overhead Push	Centroid	Centroid Controller	0	3	6
Pink Box	Overhead Push	Ellipse	Spin Compensation	2	2	5
	Gripper Sweep	Centroid	Centroid Controller	1	1	4
	Overhead Push	Sphere	Centroid Controller	1	1	6
	Overhead Push	Bounding Box	Centroid Controller	1	2	6
	Overhead Push	Bounding Box	Spin Compensation	0	3	6
	Overhead Push	Centroid	Centroid Controller	0	1	6
Shampoo	Overhead Push	Bounding Box	Spin Compensation	0	1	6
Telephone	Overhead Push	Centroid	Centroid Controller	0	3	6
	Overhead Push	Bounding Box	Spin Compensation	0	3	4
Soap Box	Overhead Push	Bounding Box	Spin Compensation	0	2	6
	Gripper Sweep	Bounding Box	Centroid Controller	0	1	6
Mug	Overhead Push	Bounding Box	Centroid Controller	2	2	5
	Overhead Push	Sphere	Centroid Controller	2	2	6
	Overhead Push	Centroid	Centroid Controller	1	2	4
	Gripper Sweep	Bounding Box	Centroid Controller	0	3	6
	Gripper Sweep	Centroid	Centroid Controller	0	1	4
	Gripper Sweep	Bounding Box	Spin Compensation	0	1	6
Medicine Bottle	Gripper Sweep	Sphere	Centroid Controller	3	3	6
	Gripper Sweep	Centroid	Centroid Controller	1	2	5
	Overhead Push	Bounding Box	Centroid Controller	1	5	6
	Gripper Sweep	Bounding Box	Centroid Controller	0	3	6
Teddy Bear	Overhead Push	Centroid	Centroid Controller	3	3	6
	Gripper Sweep	Ellipse	Spin Compensation	0	1	6
	Gripper Sweep	Bounding Box	Centroid Controller	0	1	5
Red Bottle	Gripper Sweep	Centroid	Centroid Controller	1	1	5
	Overhead Push	Centroid	Centroid Controller	0	3	6
	Gripper Sweep	Bounding Box	Centroid Controller	0	1	6
	Overhead Push	Bounding Box	Centroid Controller	0	1	6
TV Remote	Overhead Push	Ellipse	Spin Compensation	0	3	6
Bowl	Overhead Push	Ellipse	Spin Compensation	2	3	6
	Overhead Push	Bounding Box	Centroid Controller	2	3	6
	Gripper Sweep	Centroid	Centroid Controller	1	3	3
	Overhead Push	Centroid	Centroid Controller	0	1	4
	Gripper Sweep	Bounding Box	Centroid Controller	0	1	6
Salt	Overhead Push	Bounding Box	Spin Compensation	2	2	6
	Gripper Sweep	Bounding Box	Centroid Controller	2	3	4
	Overhead Push	Centroid	Centroid Controller	1	2	2
	Overhead Push	Bounding Box	Centroid Controller	1	1	4
	Gripper Sweep	Centroid	Centroid Controller	0	2	6
	Gripper Sweep	Bounding Box	Spin Compensation	0	1	6

TABLE II: Successful (final error within 0.02m or 0.04m of goal) affordance-based behaviors for each object.

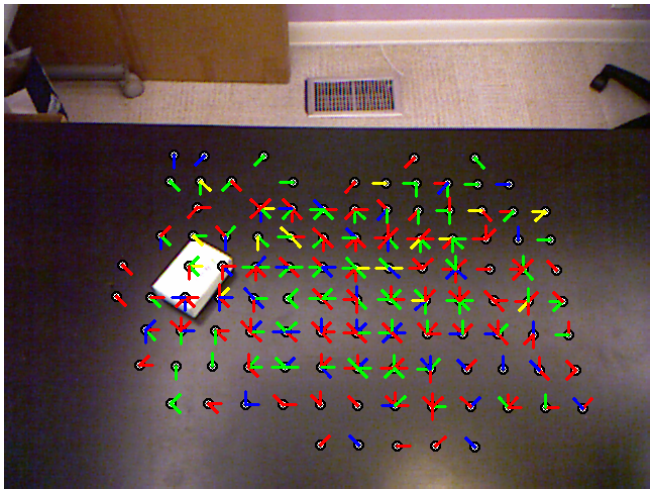

Fig. 5: Best on average performing behavior primitives for different initial locations and pushing angles. Green is gripper sweep. Blue is fingertip push. Red is overhead push. Yellow is gripper pull. Bottom of the image is closer to the robot (smaller x). Left of the image is left for the workspace (positive y).

average performance at each location and angle between the four behavior primitives in Figure 5. We note how at the edges of the workspace farthest from the robot the gripper pull behavior primitive performs best on average. The robot has discovered that at the farthest extent of its workspace the gripper pull behavior is the most effective choice it can make to push or pull an object to a desired location. We can perform similar analysis to allow the robot to automatically select between the left and right arms.

VIII. Conclusions and Future Work

We have presented a novel behavior representation by which a robot can systematically explore the affordances of objects. Our method allows us to find the most likely to succeed behavior as a function of object ID or location in the workspace. We wish to extend this method to help predict the success of behaviors for objects the robot has not had any previous experience manipulating. Additionally we wish to make more efficient use of our training data by the use of more sophisticated learning methods to better share information across objects and the workspace. We also wish to accelerate the learning process by exploring behaviors which will be most informative, instead of performing a naive exhaustive search.

References

[1] J. J. Gibson, "The theory of affordances," in *Perceiving, Acting, and Knowing: Toward an Ecological Psychology*, R. Shaw and J. Bransford, Eds. Hillsdale, NJ: Lawrence Erlbaum, 1977, pp. 67–82.

[2] F. Iida, R. Pfeifer, and L. Steels, *Embodied Artificial Intelligence International Seminar, Dagstuhl Castle, Germany, July 7-11, 2003, Revised Papers*, ser. Lecture Notes in Computer Science. Springer, 2004, vol. 3139.

[3] A. N. Erkan, O. Kroemer, R. Detry, Y. Altun, J. Piater, and J. Peters, "Learning probabilistic discriminative models of grasp affordances under limited supervision," in *IROS*, Oct. 2010, pp. 1586–1591.

[4] D. Katz, Y. Pyuro, and O. Brock, "Learning to Manipulate Articulated Objects in Unstructured Environments Using a Grounded Relational Representation," in *RSS*, Zurich, Switzerland, June 2008, pp. 254–261.

[5] J. Barry, K. Hsiao, L. P. Kaelbling, and T. Lozano-Pérez, "Manipulation with Multiple Action Types," in *International Symposium on Experimental Robotics (ISER)*, 2012.

[6] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, and G. Sandini, "Learning about objects through action - initial steps towards artificial cognition," in *ICRA*, vol. 3, Sept 2003, pp. 3140–3145.

[7] G. Metta and P. Fitzpatrick, "Early integration of vision and manipulation," *Adaptive Behavior*, vol. 11, no. 2, pp. 109–128, 2003, special Issue on Epigenetic Robotics.

[8] G. Fritz, L. Paletta, R. Breithaupt, E. Rome, and G. Dorffner, "Learning predictive features in affordance based robotic perception systems," in *IROS*, Beijing, China, Oct 2006, pp. 3642–3647.

[9] J. Sun, J. L. Moore, A. Bobick, and J. M. Rehg, "Learning Visual Object Categories for Robot Affordance Prediction," *The International Journal of Robotics Research*, vol. 29, no. 2-3, pp. 174–197, 2010.

[10] T. Hermans, J. M. Rehg, and A. Bobick, "Affordance Prediction via Learned Object Attributes," in *IEEE International Conference on Robotics and Automation (ICRA): Workshop on Semantic Perception, Mapping, and Exploration*, May 2011.

[11] M. Lopes, F. S. Melo, and L. Montesano, "Affordance-based imitation learning in robots," in *IROS*, San Diego, CA, 2007, pp. 1015–1021.

[12] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor, "Modeling object affordances using bayesian networks," in *IROS*, 2007.

[13] ——, "Learning object affordances: From sensory-motor coordination to imitation," *IEEE Trans. on Robotics*, vol. 24, no. 1, pp. 15–26, Feb 2008.

[14] M. Stark, P. Lies, M. Zillich, J. Wyatt, and B. Schiele, "Functional object class detection based on learned affordance cues," in *Sixth Intl. Conf. on Computer Vision Systems (ICVS 08)*, Santorini, Greece, May 2008, pp. 435–444.

[15] E. Ugur, E. Oztop, and E. Sahin, "Going Beyond The Perception of Affordances: Learning How to Actualize Them Through Behavioral Parameters," in *ICRA*, May 2011.

[16] O. Kroemer, E. Ugur, E. Oztop, and J. Peters, "A Kernel-based Approach to Direct Action Perception," in *ICRA*, May 2012.

[17] C. Geib, K. Mourao, R. Petrick, N. Pugeault, M. Steedman, N. Krueger, and F. W org otter, "Object Action Complexes as an Interface for Planning and Robot Control," in *Humanoids*, Genova, Italy, Dec 4-6 2006.

[18] E. Ugur, E. Sahin, and E. Oztop, "Unsupervised Learning of Object Affordances for Planning in a Mobile Manipulation Platform," in *ICRA*, May 2011.

[19] U. Klank, L. Mösenlechner, A. Maldonado, and M. Beetz, "Robots that Validate Learned Perceptual Models," in *ICRA*, May 2012.

[20] M. T. Mason, "Mechanics and Planning of Manipulator Pushing Operations," *The International Journal of Robotics Research (IJRR)*, vol. 5, pp. 53–71, September 1986.

[21] D. Omrčen, C. Böge, T. Asfour, A. Ude, and R. Dillmann, "Autonomous Acquisition of Pushing Actions to Support Object Grasping with a Humanoid Robot," in *Humanoids*, Paris, France, 2009.

[22] M. Dogar and S. Srinivasa, "A Framework for Push-Grasping in Clutter," in *RSS*, 2011.

[23] ——, "Push-Grasping with Dexterous Hands: Mechanics and a Method," in *IROS*, 2010.

[24] A. Cosgun, T. Hermans, V. Emeli, and M. Stilman, "Push Planning for Object Placement on Cluttered Table Surfaces," in *IROS*, 2011.

[25] T. Hermans, J. M. Rehg, and A. Bobick, "Guided Pushing for Object Singulation," in *IROS*, 2012.

[26] F. Ruiz-Ugalde, G. Cheng, and M. Beetz, "Fast Adaptation for Effect-aware Pushing," in *Humanoids*, 2011.

[27] S. Narasimhan, "Task Level Strategies for Robots," Ph.D. dissertation, Massachusetts Institute of Technology, 1994.

[28] M. Salganicoff, G. Metta, A. Oddera, and G. Sandini, "A vision-based learning method for pushing manipulation," in *AAAI Fall Symposium on Machine Learning in Computer Vision*, 1993.

[29] J. Scholz and M. Stilman, "Combining Motion Planning and Optimization for Flexible Robot Manipulation," in *Humanoids*, 2010.

[30] T. Hermans, J. M. Rehg, and A. F. Bobick, "Decoupling Behavior, Control, and Perception in Affordance-Based Manipulation," in *(IROS) Workshop on Cognitive Assistive Systems*, October 2012.

[31] S. Goyal, A. Ruina, and J. Papadopoulos, "Limit Surface and Moment Function Descriptions of Planar Sliding," in *ICRA*, 1989.

[32] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, pp. 381–395, June 1981.

[33] L. Y. Chang, J. R. Smith, and D. Fox, "Interactive Singulation of Objects from a Pile," in *ICRA*, 2012.