Learning and Generalization of Behavior-Grounded Tool Affordances

Jivko Sinapov
Department of Computer Science
Iowa State University
jsinapov@cs.iastate.edu

Alexander Stoytchev
Department of Computer Science
Iowa State University
alex@cs.iastate.edu

***Abstract*— This paper describes an approach which a robot can use to learn the effects of its actions with a tool, as well as identify which frames of reference are useful for predicting these effects. The robot learns the tool representation during a behavioral babbling stage in which it randomly explores the space of its actions and perceives their effects. The experimental results show that the robot is able to learn a compact and accurate model of how its tool actions would affect the position of a target object. Furthermore, the model learned by the robot can generalize and perform well even with tools that the robot has never seen before. Experiments were conducted in a dynamics robot simulator. Two different learning algorithms and five different frames of reference were evaluated based on their generalization performance.**

***Index Terms*— Developmental Robotics, Affordances, Learning of Affordances, Tool Affordances.**

I. Introduction

The term *affordance* was coined by J. J. Gibson who paid special attention in his research to environmental objects and the types of properties that living organisms perceive about them. Gibson defines affordances as action possibilities that are present in the environment and can be perceived by an individual [1]. In the context of tool use, the affordances of a tool correspond to the possible uses that the tool affords to an organism. Furthermore, Gibson also argues that humans naturally perceive affordances when they encounter environmental objects [1].

Similar observations have been made by Jean Piaget [2] who formulated one of the most influential developmental theories of the 20-th century. According to Piaget environmental objects play an important role in human development. In the early stages of development, the actions of a child are directed at objects with the goal of establishing associations between its actions and outcomes in the external world [2]. In the terminology of Gibson this would be equivalent to trying to learn the affordances of the objects. Similar mechanisms are at play when the child learns to use tools [2].

In previous work, Stoytchev [3] introduced a computational framework for a behavior-grounded representation of tools. This paper extends that model by describing an approach which a robot can use to learn the effects of its actions with a tool, as well as detect the features in its sensory stream which are useful for predicting these effects. The learned model constitutes a compact representation of the action possibilities that the tool affords the robot. The robot learns the representation by exploring the space of its actions with the tool and observing their effects. The predictive model is grounded in the robot's own perceptual and behavioral repertoire, allowing the robot to autonomously test and verify its knowledge of the tool. In addition, the framework presented in this paper allows the robot to evaluate the predictive power of different frames of reference. There is evidence that biological brains maintain multiple coordinate frames in order to coordinate bodily movements [4], [5]. Gallistel, for example, suggests that intelligent behavior is about learning how to coordinate these frames [5].

Once a model is learned, the robot is tested on how well it can predict the consequences of its actions using this model. Furthermore, the generalization abilities of the learned models are evaluated. In abstract terms, generalization can be defined as the ability to correctly estimate the values of a given function at points in its input space for which data is not available [6]. In this paper, this function is a model for the observable consequences of executing a particular action with a given tool. Different learning algorithms and frames of reference are evaluated in the course of the experiments.

II. Related Work

A. Tool Experiments with Animals

Tool-using experiments have been used for the last 90 years to test the intelligence of primates [7], [8], [9]. Wolfgang Köhler [7] conducted one of the first systematic studies of tool-using behaviors in primates. His experiments required the use of tools such as sticks, hooks, boxes, and ladders in order to complete a given task and extract a reward in the form of food.

Povinelli *et al.* [8] replicated many of the experiments conducted by Köhler and analyzed the results using statistical techniques. They concluded that chimpanzees do not reason about actions and tool tasks in abstract terms such as mass, force, or gravity. Another conclusion was that the primates solved tool-using tasks by extracting simple rules from their experience, *e.g.*, "contact leads to movement" [8].

More recently, Ishibashi *et al.* [10] conducted experiments with macaque monkeys which demonstrate the ability of primates to generalize their tool related knowledge to novel tools. Their study shows that macaque monkeys can use a novel tool provided that they have encountered a similar tool in the past. For example, once the monkey has been exposed to a T-stick tool (see Figure 2), it was also capable of performing the task with other similar tools.

The experimental setup described in Section IV was influenced by the setup used by Ishibashi *et al.* [10].

B. Tool Experiments with Robots

Our previous work [3] introduced a method for representing the affordances of tools by grounding the representation in the behavioral and perceptual repertoire of the robot. In this representation the affordances of the tool are expressed in concrete terms (*e.g.*, behaviors and observable outcomes) that are directly available to the robot. Therefore, the robot is able to directly test and verify its tool representation. This is consistent with the verification principle which states that a robot should not attempt to learn anything that it cannot verify for itself [11], [12].

The current implementation of the behavior-grounded approach [3] suffers from several limitations. First, the approach keeps the learned affordances in a lookup table and does not allow for predictions to be computed for data points that have not already been included. Second, the method does not perform any generalization across multiple tools. As a result, the exploration of novel tools starts completely from scratch, even though the new tool may be similar to an already explored tool.

This paper attempts to overcome these limitations by introducing a framework which allows the robot to learn a compact predictive model for the affordances of a tool. This model is also grounded in the robot's own perceptual and behavioral abilities but in addition to that it also has certain generalization properties.

III. Problem Statement

The learning task of the robot is described with the help of the following notation. Let A_t be the action performed by the robot at time t. Furthermore, let $S_t = [s_1, s_2, \ldots, s_n]$ be a vector of sensory inputs received by the robot at time t. The robot can extract features from its raw sensory information with the help of *perceptual functions* $\Phi_1, \ldots, \Phi_m$. These functions transform the original sensory input vector, S_t, into another vector, i.e., $\Phi_i(S_t) \rightarrow U_t$, where $U_t = [u_1, \ldots, u_k]$, and $k \ll n$.

A change detection function, Δ_f, determines if some event of interest (*e.g.*, visual movement) has occurred. The function is defined as $\Delta_f(S_{t-k}, S_t) \rightarrow C_t$ where C_t is is a vector describing the change, and S_{t-k} and S_t are the sensory input vectors at times $t-k$ and t, respectively.

The task of the robot is to learn a model M_f such that $M_f(A_t, \Phi_i(S_t)) \rightarrow \Delta_f(S_t, S_{t+k})$ for some perceptual function Φ_i. In other words, the robot needs to learn a predictive model of the future consequences (as measured by the change detection function Δ_f) of executing action A_t, given the current sensory input vector S_t which is modified by the perceptual function Φ_i.

IV. Experimental Setup

A. Simulator, Robot, and Tools

All experiments were performed using the open-source dynamic robot simulator BREVE [13]. The simulated robot, the tools, and the objects in the simulation were modeled as rigid bodies. The physics parameters of the simulator such as gravity constant and coefficient of friction were set to their default values.

The robot is a simulated arm with 6 degrees of freedom: a slider joint at the base, waist roll, shoulder pitch, elbow pitch, wrist pitch, and wrist roll. The robot also has a gripper attached to the wrist. A snapshot of the simulated robot arm is shown in Figure 1.a.

Six different tools were used in the experiments as shown in Figure 2. Each tool was colored in red. Two yellow markers were also placed on each tool (see Figure 2). The marker at the front end of the tool will be referred to as marker #1, while the marker closest to the robot arm will be referred to as marker #2. The last object in the simulation is a small cylindrical puck which can be moved by the tool when the robot performs an action.

B. Sensory Input and Frames of Reference

The robot's sensory input, S_t, is extracted from a camera positioned directly overhead and looking downward. A sample visual input can be seen in Figure 1.b. Each pixel

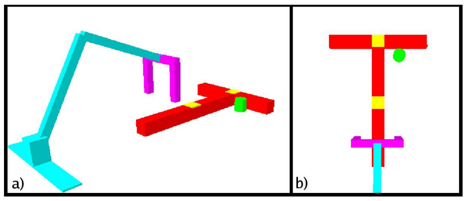

Fig. 1. a) Snapshot of the robot arm in the dynamics simulator; b) View from the robot's simulated camera.

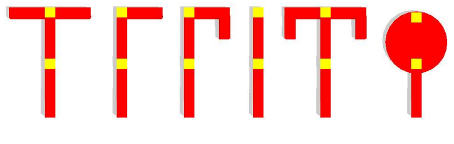

Fig. 2. The six different tools used by the robot: 1) T-stick; 2) L-stick; 3) L-hook; 4) Stick; 5) T-hook; and 6) Paddle.

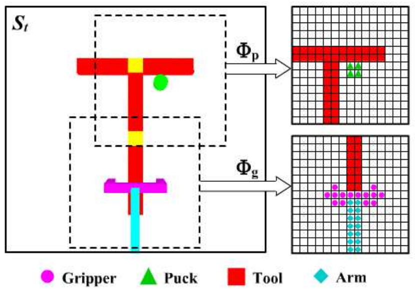

Fig. 3. A view from the robot's simulated camera and the computation of the perceptual functions $\Phi_p(S_t)$ and $\Phi_g(S_t)$ from the sensory input S_t.

in the image is labeled as belonging to either the tool, the puck, the gripper, the arm, or the background. Formally, $S_t = \{T, P, G, A, B\}^{792\times 482}$ where 792×482 is the size of the simulated camera image, and T, P, G, A, and B stand for "Tool", "Puck", "Gripper", "Arm", and "Background" respectively. The robot implicitly knows that the pixels corresponding to the two tool markers are part of the tool.

The raw sensory input is too large to be useful for the purposes of generalization. Thus, it might be beneficial for the robot to focus only on a small part of its sensory input for prediction. For example, when writing with a pen, it is more helpful to focus on its tip as opposed to some object in the background. The robot is capable of using five different frames of reference for prediction. The first of these is a reference frame centered on the camera image that the robot receives as sensory input. The other four frames of reference are centered at one of the following objects of interest: the gripper, tool marker #1, tool marker #2, or the puck.

The different frames of reference are associated with five perceptual functions: Φ_c, Φ_g, Φ_{t_1}, Φ_{t_2}, Φ_p. The functions are computed in the following way: a window of size 300×300 is centered at either the image center (Φ_c), the gripper (Φ_g), tool marker #1 (Φ_{t_1}), tool marker #2 (Φ_{t_2}), or the puck (Φ_p). The window is cropped from the image S_t and scaled down to size 15×15. The resulting image is returned as a 225-dimensional output of the perceptual function. Figure 3 shows this process for the perceptual functions Φ_p and Φ_g. The predictive power of each frame of reference is evaluated for different tools in Section VI.

C. Change Detection Function

The robot is equipped with one change detection function, Δ_θ, which tracks the direction of the puck's displacement over time. The function takes two sensory vectors, S_t and S_{t+k}, as input and returns a tuple $[D_x, D_y]$, such that D_x and D_y are the x and y displacements of the puck in the robot's field of view between times t and $t+k$, where k is set to 10 frames. This change detection functions allows the robot to detect whether the puck is displaced in a given direction as a result of the robot's action with the tool. Note that the output of the change detection function does not depend on which frame of reference the robot is using for building its model.

D. Behavioral Babbling and Data Collection

The robot collects sample data about the effects of the tool on the puck through behavioral babbling. Each robot action consists of grasping the tool at a prespecified position, and then sliding it in the horizontal plane by some given x and y offsets, which are the parameters of the action.

Each trial of the behavioral babbling process is performed as follows. First, the tool is positioned in front of the robot and the puck is placed in a random location around the tool. In each trial the starting position of the tool is offset to the left or the right of the robot by a random distance. After grasping the tool, the robot slides its gripper while holding the tool in the horizontal plane (this is the action A_t). The x and y offsets for A_t are chosen randomly from a uniform distribution. The tool movement continues for 30 frames in the chosen direction or until the movement is completed.

The sensory input of the robot, S_t, along with the outputs of the five perceptual functions are recorded for each frame. The values of the change detection function Δ_θ are also recorded if the attractor object moves as a result of the robot's action. Each recorded data entry corresponds to an event of interest that the robot has observed and constitutes a tuple of the form $\langle A_t, \Phi_i(S_t), \Delta_\theta(S_t, S_{t+k})\rangle$, where Φ_i is the perceptual function being used, and A_t is the robot's action parametrized by x and y. Figure 4 visualizes a trial with the T-stick tool. Using this data, the robot has to learn a predictive model which takes as input A_t and $\Phi_i(S_t)$ and outputs an estimate for the future consequences of the action A_t, as measured by the change detection function Δ_θ. The next section describes the learning methodology used to achieve that.

V. Learning Methodology

As it was described in the previous section, the task of the robot is to learn a model which can predict the outcomes of its tool actions. The inputs of such a model (consisting of $\langle A_t, \Phi_i(S_t)\rangle$) will be referred to as the set of *attributes* and

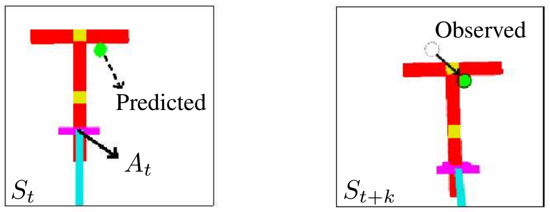

Fig. 4. A sample trial with the T-stick tool. At the start of the trial (left), the robot makes a prediction using the model M_θ regarding the outcome of its action, A_t. Once movement of the puck is detected with the change detection function Δ_θ (right), the robot is able to verify its prediction and record the observed outcome.

the output ($\Delta_\theta(S_t, S_{t+k})$) as the *class label*. Two different learning approaches were used as described below.

A. k-Nearest Neighbor

The first approach does not construct an explicit model from the training data. Instead, it stores all data points (or instances) and their class labels and only uses them when the model is asked to make a prediction. Models that have this property fall in the category of *lazy learning* or *memory-based learning* [14], [15].

The *k Nearest-Neighbor (k-NN)* algorithm used here finds the *k* closest neighbors of the new point and assigns a class label which is generally a smoothed average of the labels belonging to the *k* neighbors. The k-NN algorithm is simple and relatively robust to noise. However, without the introduction of special distance and attribute weighting functions, the model can perform poorly when there are irrelevant attributes. Nevertheless, k-NN can provide near-optimal classification and regression for many types of problems.

B. Decision Tree

The second learning approach used in this paper constructs a compact model of the data by extracting a set of structured rules (or tests). These tests can later be used to quickly classify a new instance. The classic example of such an approach is the *decision tree* [16]. A decision tree can be constructed recursively as follows: 1) start with the entire set of data instances at the root of the tree; 2) recursively add nodes to the tree by partitioning the set into subsets based on tests that yield the minimum expected entropy; 3) stop the recursion when the entropy of every leaf node is minimized.

Each non-leaf node in the decision tree contains a test on one or more of the attributes. Once the decision tree is constructed it can be used to generate simple *if-then* rules which can serve as predictors of future outcomes. In the case of numeric prediction problems, the leaf node might contain a linear regression model which is then used to compute the predicted value [17].

C. Model Evaluation

Once the robot has conducted a specified number of trials, it learns the model M_θ for the corresponding change detection function Δ_θ. The implementations of the decision tree and k-NN models included in the WEKA open source machine learning package [18] were used in this work. The models were evaluated by performing *3-fold cross-validation* on the recorded data. The set of trials was split into three disjoint sets and at each iteration of the evaluation process the data from two of these sets was used for training the model and the data from the remaining set was used for evaluation. To assess the performance of the M_θ model, the angle of the movement direction of the puck was compared to the angle of the predicted direction. A prediction was considered good if the error (in terms of angular degrees) was less than 20°. The predictive power of each of the five perceptual functions (Φ_c, Φ_g, Φ_{t_1}, Φ_{t_2}, and Φ_p) was evaluated using the two learning approaches (i.e., k-NN and decision tree) as described in the next section.

TABLE I
PERCENTAGE OF GOOD PREDICTIONS FOR Δ_θ ACROSS DIFFERENT FRAMES OF REFERENCE (DECISION TREE ALGORITHM)

Tool	Frame of Reference				
	Φ_c	Φ_g	Φ_{t_1}	Φ_{t_2}	Φ_p
T-stick	51.3%	44.1%	71.4%	57.6%	**94.4%**
L-stick	40.6%	52.1%	78.8%	68.6%	**92.7%**
L-hook	47.9%	50.7%	67.3%	56.1%	**87.2%**
Stick	72.1%	69.3%	86.2%	85.8%	**90.2%**
T-hook	63.4%	43.2%	57.4%	49.4%	**88.1%**
Paddle	41.7%	50.5%	59.4%	53.7%	**85.2%**

TABLE II
PERCENTAGE OF GOOD PREDICTIONS FOR Δ_θ ACROSS DIFFERENT FRAMES OF REFERENCE (K-NN LEARNING ALGORITHM)

Tool	Frame of Reference				
	Φ_c	Φ_g	Φ_{t_1}	Φ_{t_2}	Φ_p
T-stick	28.6%	37.5%	**88.8%**	59.5%	80.9%
L-stick	27.3%	47.5%	**90.4%**	56.7%	82.8%
L-hook	21.4%	36.7%	**80.0%**	53.6%	76.0%
Stick	42.1%	54.7%	**85.0%**	69.3%	79.1%
T-hook	27.8%	32.9%	52.5%	44.6%	**72.4%**
Paddle	22.2%	31.8%	**68.5%**	40.6%	63.6%

VI. EXPERIMENTAL RESULTS

A. Exploring a Single Tool

The data for the first set of experiments consists of 600 exploratory trials with each of the six tools for a total of 3000. In about half of the trials the action has no effect on the puck, and therefore the robot was first evaluated on whether it can predict if its behavior would impact the puck at all. For all six tools, the accuracy rates of these predictive models were in the range of 90-95%, indicating that this task is relatively easy. The robot was then evaluated on whether it can predict the displacement direction of the puck in the cases where the puck moves as a result of the action. Tables I and II show the performance of the models learned for Δ_θ using decision tree with linear regression nodes at the leaves and k-NN.

The k-NN model performs best when using a tool-centric frame of reference (Φ_{t_1}) while the decision tree model achieves best performance with the puck-centric frame of reference (Φ_p). One possible explanation for this is that the decision tree model can handle irrelevant attributes quite well - a closer look at the produced trees reveals that the prediction rules take into account only the pixels that are close to the puck. Because the output of Φ_p is centered at the puck, the task of the decision tree model is made even easier, since the

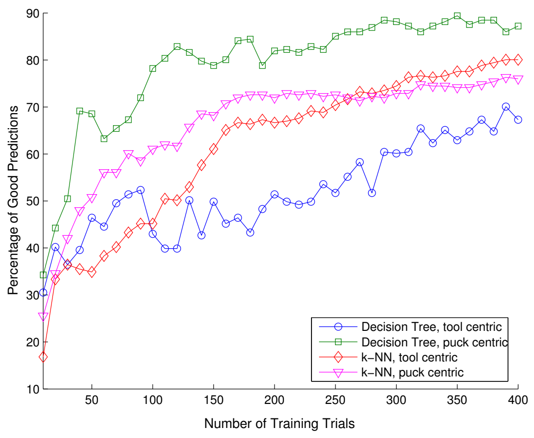

Fig. 5. Comparison between the decision tree and k-NN models for the T-hook tool, with tool centric (Φ_{t_1}) and puck-centric (Φ_p) frames of reference as robot gains more experience. The performance rates indicate the percentage of predictions whose absolute error is less than 20°.

puck's pixels are guaranteed to be in the center. The standard k-NN model used here, however, is not capable of explicitly detecting irrelevant attributes. The outputs from Φ_{t_1} contain less variance since the pixels corresponding to the tool do not change and therefore the only factor that plays a role in deciding the neighbors of a new data point is the position of the puck relative to the tool.

To determine how much experience is needed by the robot in order to learn an accurate model, another experiment was performed in which the number of training trials was varied from 10 to 400, while the number of test trials was kept constant at 200. Figure 5 shows the results for the T-hook tool. The plot shows that after less than 50 training trials the best performance is achieved when using the decision tree model with a puck-centered frame of reference.

It is worth examining the situations in which the models make prediction errors. Figure 6 visualizes the errors made by the decision tree model using a camera-centric (Φ_c) and a puck-centric (Φ_p) frame of reference for the T-hook tool. The errors of the model using Φ_c are distributed fairly uniformly around the tool. However, when using the puck-centric frame of reference, the likelihood of prediction error is higher if the puck is located near one of the tool's corners and lower if the puck is near a smooth side of the tool at the start of each trial. The results for the other five tools also show a similar pattern.

B. Generalization Across Tools

Another question of interest is whether the robot can learn from one tool and apply that knowledge to predict the affordances of another tool. An experiment was conducted during which the robot was given a single tool to explore but then evaluated on a different one. Table III shows the performance

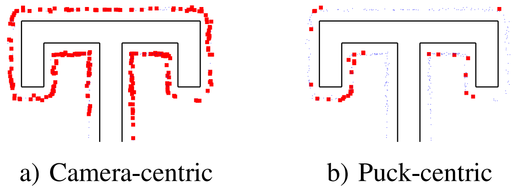

Fig. 6. Visualization of the prediction errors made by the decision tree model with camera-centric (left) and puck-centric (right) frame of reference for the T-hook tool. Each point in the plot (red or blue) represents the puck's starting position relative to the tool during some particular trial. The points represented by the large red squares indicate cases in which the prediction error is greater than 20°.

TABLE III

MODEL PERFORMANCE FOR Δ_θ (DECISION TREE ALGORITHM)

Train Tool	Test Tool	Frame of Reference				
		Φ_c	Φ_g	Φ_{t_1}	Φ_{t_2}	Φ_p
T-stick	T-hook	42.6%	37.1%	49.5%	42.9%	**86.6%**
T-stick	Paddle	34.6%	35.8%	24.7%	28.2%	**76.7%**
T-hook	L-stick	45.2%	37.9%	48.8%	41.5%	**85.4%**
L-hook	T-hook	50.5%	43.7%	59.6%	47.2%	**80.0%**

of the decision tree models for several such combinations. The results obtained when using the k-NN algorithm for this experiment (not shown due to space limitations) were substantially worse across all five frames of reference and indicate that the memory-based learning model has a hard time generalizing across new tools. When using a decision tree model, the robot is able to best predict the affordances of the new tool when using a frame of reference centered at the puck. This comes as no surprise, since the local neighborhood of the puck does not vary as much across multiple tools.

The results also show that it is easier to generalize across tools which have similar local features. For example, the model learned from experience with the T-stick is also useful in predicting the affordances of the T-hook. However, models learned from experience with any of the first five tools do not perform as well when using the Paddle, a tool which is unique in its circular shape. This is consistent with observations of monkey tool use conducted by Ishibashi *et al.* [10], which show that the macaques preferred to use tools similar to the T-hook to which they were exposed and were not capable of retrieving food with novel circular-shaped tools.

C. Short and Long Tool

In this experiment, the robot is given a tool to explore and then tested with a larger version of the same tool. Ishibashi *et al.* [10] showed that monkeys are generally able to use larger or smaller versions of tools to which they have been exposed previously. The tool chosen for this experiment was the L-hook, and after the robot has explored it, the tool's

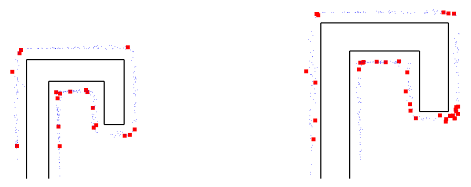

Fig. 7. Visualization of the prediction errors made by decision tree model using a puck-centered frame of reference for the regular (left) and the enlarged (right) L-hook tools. In both scenarios, the model is trained on the regular-sized tool. Each point in the plot (red or blue) represents the puck's starting position relative to the tool during some particular trial. The points represented by the large red squares indicate cases in which the prediction error is greater than 20°.

dimensions were increased in size by 33%. The performance rates (measured as percentage of good predictions) of the decision tree models for the five different frames of reference were 35.5% (Φ_c), 35.2% (Φ_g), 56.4% (Φ_{t_1}), 30.9% (Φ_{t_2}) and 83.8% (Φ_p). Thus, the predictive performance when using a puck-centered frame of reference (Φ_p) is still good. Figure 7 visualizes the locations of the prediction errors and shows that the model trained on the small L-hook performs similarly well when evaluated on the larger version of the tool. In both cases the errors are clustered around the corners of the tool.

The k-NN learning algorithm was also evaluated on this task, but the results were significantly worse across all five perceptual functions. The best rate of 56.1% was achieved when using Φ_p. This comparisson shows that the compact decision tree model is much better at generalizing in this situation than the memory-based k-NN algorithm.

VII. Conclusion and Future Work

This paper presented a framework for learning behavior-grounded tool affordances with generalization across multiple tools. Three sets of experiments were conducted in which the robot was tested on predicting the affordances of familiar tools, novel tools, and larger versions of familiar tools. Two learning algorithms (k-NN and decision tree) were evaluated on these tasks and the results show that the compact decision tree model significantly outperforms the k-NN algorithm. The frame of reference centered on the attractor object contains the most predictive information for the given tool task. Generalization across tools is best achieved when the novel tool shares local features with one of the tools to which the robot has been previously exposed.

The framework also allows for the prediction errors to be quantified. As was shown previously, the chance of making a prediction error depends on the position of the puck relative to the tool. If the puck is located near the corner of a tool, the chances of making a prediction error are much higher.

There are several natural extensions to this model to be addressed in future work. First, the proposed model could be integrated in a planing system that solves tool-using tasks. The decision tree and k-NN learning models have natural inverse models which could be used by the robot when planing an action with the tool in order to achieve a certain goal. Furthermore, by quantifying the errors of the models, the robot would be able to select ways to use the tool which minimize the uncertainty in predictions in the course of planning and action execution.

Another problem to be resolved is that of model selection. The results from this paper show that it is not necessary to start the exploration of a new tool from scratch. However, if the robot has been exposed to multiple tools and has learned a model for each one of them, it would need to autonomously select which model to use when it encounters a novel tool.

References

[1] J. J. Gibson, *The ecological approach to visual perception*. Boston: Houghton Mifflin, 1979.

[2] J. Piaget, *The origins of intelligence in the child*. New York: Norton, 1952.

[3] A. Stoytchev, "Behavior-grounded representation of tool affordances," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2005, pp. 3071–3076.

[4] N. S. Newcombe and J. Huttenlocher, *Making Space: The Development of Spatial Representation and Reasoning*. Cambridge, MA: MIT Press, 2000.

[5] C. R. Gallistel, "Coordinate transformations in the genesis of directed action," in *Cognitive science*, B. O. M. Bly and D. E. Rummelhart, Eds. New York: Academic, 1999, pp. 1–42.

[6] T. Poggio and E. Bizzi, "Generalization in vision and motor control," *Nature*, vol. 431, pp. 768–774, 14 October 2004.

[7] W. Köhler, *The mentality of apes*. Harcourt, Brace, and Co., 1931.

[8] D. Povinelli, J. Reaux, L. Theall, and S. Giambrone, *Folk Physics for Apes: The Chimpanzee's theory of how the world works*. Oxford Univ. Press, 2000.

[9] E. Visalberghi and L. Trinca, "Tool use in capuchin monkeys: distinguishing between performing and understanding," *Primates*, vol. 30, pp. 511–21, 1989.

[10] H. Ishibashi, S. Hihara, and A. Iriki, "Acquisition and development of monkey tool-use: behavioral and kinematic analysis," *Canadian J. of Physiology and Pharmacology*, vol. 78, pp. 958–966, 2000.

[11] R. Sutton, "Verification," 2001, on-line essay http://www.cs.ualberta.ca/˜sutton/IncIdeas/Verification.html.

[12] A. Stoytchev, "Five basic principles of developmental robotics," in *NIPS 2006 Workshop on Grounding Perception, Knowledge and Cognition in Sensori-Motor Experience*, 2006.

[13] J. Klein, "BREVE: a 3D environment for the simulation of decentralized systems and artificial life," in *ICAL 2003: Proceedings of the eighth international conference on Artificial life*. Cambridge, MA, USA: MIT Press, 2003, pp. 329–334.

[14] W. D. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithm," *Machine Learning*, vol. 6, pp. 37–66, 1991.

[15] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning," *Artificial Intelligence Review*, vol. 11, no. 1-5, pp. 11–73, 1997.

[16] J. R. Quinlan, *C4.5: programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[17] Y. Wang and I. Witten, "Inducing model trees for continuous classes," in *Proceedings of poster papers, 9th European conference on Machine Learning*, Prague, Czech Republic, 1997.

[18] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.