Effective Transfer Learning of Affordances for Household Robots

Chang Wang, Koen V. Hindriks and Robert Babuska

Abstract—Learning how to use functional objects is essential for robots that are to carry out household tasks. However, learning every object from scratch would be a very naive and time-consuming approach. In this paper, we propose transfer learning of affordances to reduce the number of exploratory actions needed to learn how to use a new object. Through embodied interaction with the object, the robot discovers the object’s similarity to previously learned objects by comparing their shape features and spatial relations between object parts. The robot actively selects object parts along with parameterized actions and evaluates the effects on-line. We demonstrate through real-world experiments with the humanoid robot NAO that our method is able to speed up the use of a new type of garbage can by transferring the affordances learned previously for similar garbage cans.

I. INTRODUCTION

Humans are able to learn the use of household objects by following instructions, imitating others or simply by trial and error. Some of the learned knowledge associates objects with particular actions, e.g., pulling handles to open doors, turning valves to control the water flow, etc. The concept of affordance [2] addresses such action possibilities for a human in a given environment [10]. In cognitive robotics, affordances have been modeled as relations between objects, actions, and effects [7] and formalized as object-action complexes [5]. Much research has been done into the learning of basic affordances, such as rollability [1], liftability [8], or pushability [4]. The key benefit of affordances is that they provide information about the effects of actions on objects. This information can then be stored and reused in a range of tasks. In the literature, learned affordances have been employed in action planning [17], imitation [7], object recognition [14], and tool use [13], [15]. However, the acquisition of affordances is often bound to the “motor babbling” [1], an exercise based on random exploratory actions performed with sole purpose of collecting training data. In this paper, we follow a different approach: affordances are learned and used during the execution of multiple tasks.

In our previous research [19], we proposed an architecture for simultaneous learning and use of affordances in goal-directed tasks. However, the learned affordances were used for specific learning tasks and objects. In this paper, we extend the approach toward long-term learning and use of affordances across a spectrum of tasks and objects.

In the literature, it has been almost exclusively assumed that discrete robot actions are always effective for manipulating objects in well designed environments. However, this is generally not true in household settings where discrete actions easily fail due to the uncertainty and variability of the environment. Successful manipulation in such settings requires the use of parameterized actions (i.e., actions defined in continuous spaces), where the robot must learn appropriate values of the action parameters [15], [18].

We combine the concepts of transfer learning [16] and affordances to structure the search for action parameters through taking advantage of previous experience with successful handling of similar objects. In this way, learning in a target task can benefit from the reuse of learned affordances in related source task(s). The main contributions of this paper are:

- We develop a formal affordance learning model for autonomous robot use of household objects that are composed of several parts.
- We propose a transfer learning architecture for learning parameterized continuous actions.
- The methods are validated in real-world experiments with NAO robot for garbage can manipulation.

The paper is organized as follows: Section II describes the affordance model, followed by the affordance learning and use approaches in Section III. Section IV proposes the transfer learning method. Section V introduces the task environment for evaluation and the experimental results. Section VI concludes the paper and outlines our plans for future work.

II. AFFORDANCE MODEL

As in our previous research [19], we define an affordance as the following triple:

\[(Object, Action, Effect)\] \hspace{1cm} (1)

Object refers to a household object and/or its part, e.g., a garbage can with a lid and a handle. Action refers to a repertoire of motor skills that can be used to interact with the object, e.g., push, or lift. Effect refers to the outcome of applying the action to the object, e.g., the handle is displaced, or the lid is open (see Fig. 1). Throughout the paper, we use the lid opening task illustrated in this figure as a running example. We start by extending affordance definition (1) to the case of affordance learning for objects composed of several parts.

A. Perception of objects, their parts and states

The robot perceives its environment and extracts visual features from its camera image. We assume that the robot can identify object parts based on known features (markers
in our experiments) and color segmentation to retrieve connected components of 2D pixels called blobs [3]. Each blob corresponds to an object part and is bounded by a minimum-enclosing rectangle (see Fig. 1). Based on a segmented blob and its bounding box, we use the following shape descriptors to describe each object part: area (ratio between pixel number of the blob and pixel number of the image), rectangleness (ratio between the short side and long side of the bounding box) and squareness (ratio between blob area and bounding box area). The robot then recognizes a household object as a combination of its parts, without necessarily knowing which of them are essential for the given task. Refer to [3] for more 2D shape descriptors, and refer to [12] for a more advanced method to recognize object parts with a RGB-D camera.

Denote by $O$ the set of objects, by $\Psi$ the set of all known object parts (container, lid, handle, pedal, etc.). As not all objects necessarily contain the same parts, we denote by $\Psi_o \subseteq \Psi$ the set of parts that object $o \in O$ is composed of. For an object $o$ and its part $\psi \in \Psi_o$, we use $s_o \in S_o$ and $s_\psi \in S_\psi$ to denote the state of the object and the part, respectively. The states change with time and are continuously measured by robot’s sensors. In our case, $s_o$ is the current size of the garbage can opening, and $s_\psi$ is the current position of the lid or handle. The time index is omitted for the ease of notation.

B. Robot Actions

Denote by $a(\theta)$ an action parameterized by a real parameter vector $\theta$, where $a \in \mathcal{A}$ indicates the type of action (push, pull, lift, etc.) and $\theta = (x, y, z)^T \in \Theta \subseteq \mathbb{R}^3$ is the position change of the robot’s end-effector in 3D space. The robot always approaches the vicinity of an object’s part before interacting with it.

C. Effects of Actions

The effects of action $a(\theta)$ on part $\psi \in \Psi_o$ and object $o$ are denoted by $e_\psi \in E_\psi$ and $e_o \in E_o$, respectively. They are measured by

$$e_\psi = m_1(s_\psi, s'_\psi)$$ \hspace{1cm} (2)

and

$$e_o = m_2(s_o, s'_o)$$ \hspace{1cm} (3)

where $s'_\psi$ and $s'_o$ are the states of $\psi$ and $o$ after $a(\theta)$ was applied, $m_1$ and $m_2$ are suitable metrics. For example, lifting a handle results in the displacement of the handle as well as the change of the opening size (see Fig. 1(b)). In this paper, the effect spaces are $E_\psi \subseteq \mathbb{R}$ and $E_o \subseteq \mathbb{R}$ and $m_2$ simply is subtracting the previous state from the new state:

$$e_o = s'_o - s_o$$ \hspace{1cm} (4)

III. AFFORDANCE LEARNING AND USE

In this paper, we assume that the robot already has a plan for executing a task with a specific household object. The task plan consists of an initial condition and a termination condition. For example, in the case of lid opening, a lid is initially closed and has to be opened wide enough. Following the literature [7], [17], [19], we define affordance learning as the process of learning prediction models of action effects on objects. The learned affordances can be used to select functional parts and to plan actions to handle objects in goal-directed tasks (see Table I).

A. Affordance learning

1) Prediction models: In order to learn affordances of each part $\psi$ of an object $o$, the robot needs to learn both the effect of performing an action on a part and also what this effect does to the object’s state. First, a prediction model
TABLE I
AFFORDANCE LEARNING AND USE BASED ON RELATIONS BETWEEN OBJECT (P)ARTS, (A)CTIONS, AND (E)FFECTS.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P, A)</td>
<td>E</td>
<td>Prediction model learning</td>
</tr>
<tr>
<td>E</td>
<td>(P, A)</td>
<td>Part selection and action planning</td>
</tr>
</tbody>
</table>

\(g_o\) maps from the parameterized action space \(A \times \Theta\) to the effect space \(E_\psi\) for each part \(\psi \in \Psi_o\):

\[
g_o : A \times \Theta \rightarrow E_\psi
\]  

(5)

Besides, a mapping \(h_a\) models the relation between the part and object effect spaces \(E_\psi\) and \(E_o\):

\[
h_a : E_\psi \rightarrow E_o
\]  

(6)

In this way, the model captures the functional relationships between the object and its parts.

2) Data collection: Whenever an effective interaction between the robot and an object part \(\psi\) took place, the following data tuple is collected:

\[
(o, s_o, \psi, s_\psi, a, \theta, e_\psi, e_o)
\]  

(7)

where \(o \in O\), \(\psi \in \Psi_o\) and \(a \in A\) take discrete values, while \(s_o \in S_o\), \(s_\psi \in S_\psi\), \(\theta \in \Theta\), \(e_\psi \in E_\psi\) and \(e_o \in E_o\) take values in continuous spaces.

In an exploration stage, an action of type \(a\) is typically carried out maximally \(n\) times on a given part \(\psi \in \Psi_o\) with various parameters selected from \(\Theta_{exp} \subset \Theta\). The learned model (5) with the domain of \(\Theta_{exp}\) is allowed to be generalized to the domain of \(\Theta\) for effect prediction. Besides, data collection for a pair \((\psi, a)\) will stop if the learned model is accurate enough to predict the effect of \(a\) on \(\psi\). Denote by \(D_o,\psi, a(k), k \leq n\) the set of collected data (7) after having performed \(a(\theta_1), ..., a(\theta_k)\) on \(\psi\).

3) Function approximation: By using \(D_o,\psi, a(n)\), the prediction models (5) and (6) can be learned as:

\[
e_\psi = \hat{g}_o(a, \theta)
\]  

(8)

and

\[
e_o = \hat{h}_a(e_\psi)
\]  

(9)

where \(\hat{g}_o\) and \(\hat{h}_a\) are the approximation of \(g_o\) and \(h_a\). Standard regression methods, e.g., linear regression [11] or Gaussian Process Regression [9], can be used to obtain \(\hat{g}_o\) and \(\hat{h}_a\).

B. Affordance use

1) Goal state: Denote by \(s^g_o \in S_o\) the goal object state. We assume that \(s^g_o\) is known and a specific action that can achieve \(s^g_o\) exists. In a given task, \(s^g_o\) can be hard coded or provided to the robot by demonstration [17]. The task is terminated by the following condition:

\[
||s_o - s^g_o|| \leq \delta_1
\]  

(10)

where \(s_o \in S_o\) is the current object state, \(||.||\) is the distance metric in state space \(S_o\), and \(\delta_1\) is a positive real value.

2) Part selection and action planning: Given an initial object state \(s^0_o\), the robot aims to select a part \(\psi^*\) and an action \(a^*(\theta^*)\) to achieve the goal state \(s^g_o\). Combining (4), (9) and (10), the following condition should be satisfied:

\[
||\hat{h}_a(e_\psi) - (s^g_o - s^0_o)|| \leq \delta_1
\]  

(11)

First, the robot selects a pair of part and action type \((\psi^*, a^*)\) along with the associated \(e_\psi\) which are most likely to result in (11):

\[
(\psi^*, a^*, e^*_\psi) = \arg\min_{\psi \in \Psi_o, a \in A, e_\psi \in E_\psi} ||\hat{h}_a(e_\psi) - (s^g_o - s^0_o)||
\]  

(12)

Then, the action parameter \(\theta^*\) is selected which corresponds to the minimal change of the end-effector in the action parameter space:

\[
\theta^* = \arg\min_{\theta \in \Theta} \{||\theta|| \mid ||\hat{g}_o(a^*, \theta) - e^*_\psi|| \leq \delta_2\}
\]  

(13)

where \(\delta_2\) is a positive real value. The above process of affordance learning and use is summarized in Algorithm 1.

Algorithm 1 Affordance learning and use for an object.

**Input:** A set of object parts \(\Psi_o\), an initial object state \(s^0_o\) and a goal object state \(s^g_o\);

**Output:** \(\psi^* \in \Psi_o\), \(a^* \in A\), \(\theta^* \in \Theta\);

1: for all \(\psi \in \Psi_o\) do
2: for all \(a \in A\) do
3: \(k = 1;\)
4: while \(k \leq n\) and (8) is not accurate do
5: \(\theta_k \in \Theta_{exp}\) randomly, apply \(a(\theta_k)\) on part \(\psi\), observe effects and collect data sample (7);
6: Learn (8) and (9) based on data \(D_o,\psi, a(k)\);
7: \(k \leftarrow k + 1;\)
8: end while
9: end for
10: end for
11: Select \(\psi^*\) and \(a^*\) using (12);
12: Select \(\theta^*\) using (13);

Finally, \(a^*(\theta^*)\) is performed on \(\psi^*\). If (10) is satisfied, the learning is considered a success; otherwise, the learning continues by collecting more data samples (7) or terminates after a certain number of actions.

IV. TRANSFER LEARNING OF AFFORDANCES

In this section, we discuss transfer learning of affordances from source task(s) to a target task. For the ease of description, the term target object is used for the object being learned in the target task, and source object(s) refers to the object(s) having been learned in the source task(s). Denote by \(o_{tar}\), the target object, and \(O_{src}\) the set of source objects.

A. Why to transfer

Affordance learning in Algorithm 1 involves robot interaction with every object part \(\psi \in \Psi_{o_{tar}}\). It usually takes a long time to obtain sufficient data to learn prediction models (8) and (9) before achieving the task goal (10). The robot may
benefit from initializing the interaction with $o_{tar}$ by making reference of information on how $O_{src}$ were used in the past.

The effectiveness of transfer methods depends on the relationship between the target learning problem and the source learning experiences. In this paper, we use the concept of similarity to estimate whether the target object is sufficiently related to one of the source objects. If they are related, the learning performance can be improved through transfer learning; otherwise, the learning performance would be no better or even worse than without transfer.

The aim is to achieve the goal state (10) while reducing the required number of exploratory actions:

$$N(o_{tar}) = \sum_{\psi \in \Psi_{o_{tar}}, a \in A} |D_{o_{tar}, \psi, a}(n)|$$

where $|\cdot|$ is the Cardinality of a set, and $D_{o_{tar}, \psi, a}(n)$ is the data set of applying action $a$ on part $\psi$.

B. Architecture

In the sequel, we develop a framework for learning target object manipulation through exploration of the action space and through the transfer of affordances from source objects. The main challenge is to have the robot decide by itself whether or not the affordances learned in the previous tasks can contribute to the current learning problem.

The overall architecture for transfer learning of affordances is illustrated in Fig. 2.

![Fig. 2. An architecture for transfer learning of affordances.](image)

The Task Plan provides the initial and termination conditions for the learning task, e.g., the initial and goal states for the lid opening task. Before taking any actions, the robot first observes $o_{tar}$ and obtains a set of its parts $\Psi_{o_{tar}}$. Then, it queries the database of manipulation experiences and selects a set of similar objects $O_{sim} \subset O_{src}$ those of which have functional parts that are relevant to $\Psi_{o_{tar}}$. The measure of similarity is based on features such as color, shape, size, structure, etc (see Section IV-C).

The Affordance Learning module learns the association of objects, actions and consequent effects, as detailed in Section III-A. After having collected some data about $o_{tar}$, the Transfer Assessment module evaluates whether $o_{src}$ is a good source of transfer by checking actual action effects. This prevents the robot from following a wrong action selection policy for manipulating $o_{tar}$ (see Section IV-D).

C. When to transfer

Algorithm 2 describes how relevant source objects are selected by comparing object features.

**Algorithm 2** Find relevant source objects.

**Input:** A set of source objects $O_{src}$ and a target object $o_{tar}$. An initial object state $s_{o_{src}}^0$ and a goal state $s_{o_{tar}}^g$.

Learned models $e_o = \hat{h}_a(\psi)$, where $o \in O_{src}, \psi \in \Psi_o, a \in A$.

**Output:** A subset of objects $O_{sim} \subset O_{src}$ sorted by similarity with $o_{tar}$.

1: Initialize $O_{sim} = \emptyset$;
2: Observe $o_{tar}$ and obtain its parts $\Psi_{o_{tar}}$ and their states;
3: for all $o \in O_{src}$ do
4:   $s_{o_{tar}}^0 \leftarrow s_{o_{tar}}^0, s_{o_{tar}}^g \leftarrow s_{o_{tar}}^g$;
5:   Find the functional part $\psi^* \in \Psi_o$ using (12);
6:   if $\psi^*$ is similar with $\psi \in \Psi_{o_{tar}}$ then
7:      Add $o$ to $O_{sim}$;
8: end if
9: end for
10: if $O_{sim} \neq \emptyset$ then
11:   Sort $O_{sim}$ by the similarity with $\Psi_{o_{tar}}$ based on spatial relations between object parts;
12: end if

Lines 2 through 8 search for related source objects based on whether $o_{tar}$ has a part that is similar with a functional part of an object $o \in O_{src}$. The intuition is that a functional part of a source object is probably still a functional part of the target object. In order to find the functional parts of source objects, a virtual task rehearsal is carried out for all source objects by setting their initial and goal states to the current values of the target object (Line 4). Then, a functional part $\psi^*$ for achieving the given goal is found using (12). At this point, the robot searches for a similar part with $\psi^*$ in $\Psi_{o_{tar}}$. If there exists one, $o$ is considered a potentially relevant source object for $o_{tar}$, and is added to $O_{sim}$ (Line 7). The measure of part similarity is based on the color and shape features introduced in Section II-A. The X-means algorithm can be used to cluster object parts as done in [7]. This allows generalization to novel parts that have not been seen before.

If $O_{sim} \neq \emptyset$, Line 11 sorts the objects in $O_{sim}$ by comparing object similarity based on spatial relations between object parts. The reason for using relational information is that the position of a part is typically designed for a particular type of use. For example, a top lid is likely to be lifted up or pushed down, while a side lid is likely to be pulled back or pushed forward. The relation between any two parts $\psi_i$ and $\psi_j$ is measured by their relative position in the image. In our case, denote by $above(\psi_i, \psi_j)$ the relation “part $\psi_i$ is above part $\psi_j$”. Then, the similarity between two objects is measured by counting such relations between the parts. Refer to [6] for more details.

For example, assume that $o_{tar}$ has a lid and a handle. Then a source object with the same two parts is more similar than another object with only one of the parts.
If the output of Algorithm 2 is an empty set, the robot does not transfer affordances learned previously. Otherwise, the transfer process takes place as detailed in the next section.

D. What and how to transfer

We now introduce the Transfer Assessment module of Fig. 2, whose task is to speed up affordance learning. Algorithm 3 discusses what affordances learned for the objects in \( O_{sim} \) can be transferred to the learning for object \( o_{tar} \).

Algorithm 3 Transfer learned affordances across objects.

Input: A set of source objects \( O_{src} \) and a target object \( o_{tar} \); An initial object state \( s^1_{otar} \) and a goal state \( s^2_{otar} \); Learned models \( e_o = \hat{g}_o(a, \theta) \) and \( e_o = \hat{h}_a(e_o) \), where \( o \in O_{src}, \psi \in \Theta_o, a \in A; k = 1; \)

Output: \( \psi^* \in \Psi_{otar}, a^* \in A, \theta^* \in \Theta; \)

1: Select \( o_{src} \in O_{sim}; \)
2: Select \( \psi^* \) and \( a^* \) using (12);
3: Select \( \theta^* \) for \( o_{src} \) using (13);
4: \( \theta_1 = \theta^*; \)
5: while \( k \leq n \) and \( \hat{g}_{o_{src}}(a^*, \theta) \) is not accurate do
6: Apply \( a^*(\theta_k) \) on \( \psi^* \) of \( o_{tar} \) and observe effect;
7: Check goal condition (10);
8: Learn (8) and (9) based on data \( D_{otar, \psi^*, a^*}(k) \);
9: \( \theta_k = \lambda_k \theta_k + \lambda_k \nabla \hat{g}_{o_{src}}(a^*, \theta_k); \)
10: \( k \leftarrow k + 1; \)
end while
12: Use \( \hat{g}_{o_{tar}}(a^*, \theta) \) and check goal condition (10);
13: for all \( a \in A \setminus \{a^*\} \) do
14: Randomly select \( \hat{\theta} \in \Theta_{exp} \), apply \( a(\hat{\theta}) \) on \( \psi^* \) and observe effect;
15: if the observation is inconsistent with \( \hat{g}_{o_{src}}(a, \theta) \) then
16: Collect more data and learn (8) and (9) for \( (\psi^*, a); \)
17: Use \( \hat{g}_{o_{tar}}(a^*, \theta) \) and check goal condition (10);
end if
19: end for
20: for all \( \psi \in \Psi_{o_{src}} \cap \Psi_{o_{tar}} \setminus \{\psi^*\} \) do
21: for all \( a \in A \) do
22: Repeat Line 14 through 18 substituting \( \psi^* \) with \( \psi; \)
end for
24: end for
25: Use Algorithm 1 to learn \( \Psi_{o_{tar}} \setminus \{\Psi_{o_{src}} \cap \Psi_{o_{tar}}\}; \)

Assume that the robot has already found a set of similar objects by Algorithm 2. Before taking any actions, the robot first selects a most likely source object \( o_{src} \in O_{sim} \) as a learning reference (Line 1).

Lines 2 through 4 select the functional part \( \psi^* \), associated action type \( a^* \) and parameter \( \theta^* \) to initiate the action selection for \( o_{tar} \). Lines 5 through 11 explore the parameter space of \( a^* \), which would terminate if the maximal number of action had been selected or the prediction model (8) is accurate, similar to Lines 4 through 8 in Algorithm 1. Line 7 checks whether the goal condition is achieved; if so, the learning terminates. This is likely to happen when \( o_{src} \) has similar parts with \( o_{tar} \) (see Fig. 1(a) and Fig. 1(c)). Line 9 makes use of the source object’s model of action effect on part that provides a gradient direction for searching effective action parameters. Parameter \( \lambda_k > 0 \) is a step size that guarantees \( \theta_k \in \Theta \). Line 12 uses the learned model to select an action parameter and check the goal condition as in Line 7.

If the goal is not achieved, Lines 13 through 19 continue learning \( \psi^* \) with other actions. For each action type \( a \), Line 14 selects only one action parameter and checks whether \( \hat{g}_{o_{src}}(a, \theta) \) is consistent with the new observation. As \( a \neq a^* \) proved not effective for \( \psi^* \in \Psi_{o_{src}}, \) model \( \hat{g}_{o_{src}}(a, \theta) \) predicts that \( a \) is also not effective for \( \psi^* \in \Psi_{o_{tar}} \). Otherwise, if the prediction is not correct, then the chosen action type is probably effective for \( \psi^*. \) For example, the handles in Fig. 1(b) and Fig. 1(e) are lift-able and pull-able, respectively. Other action types such as sliding left or right are ineffective for both of them. Assume \( o_{tar} \) in Fig. 1(b) has been learned using Algorithm 1. Then, each ineffective action type is only needed to be tried once to confirm for \( o_{tar} \) in Fig. 1(e). When the handle of \( o_{tar} \) is pulled, the displacement of the handle is found inconsistent with the model of \( o_{tar} \). Then, more pulling parameters are selected to learn the pull affordance.

In this way, the robot exploits the learned affordances from the source object by using not only successful experience (Lines 2 through 11), but also the information about non-functional parts (Lines 13 through 19).

Till Line 20, the functional part \( \psi^* \in \Psi_{o_{src}} \) has been proved not functional for \( o_{tar} \). Lines 20 through 24 check each of the learned parts with each action type in the same way, as do Lines 14 through 18. Finally, Line 25 uses Algorithm 1 to interact with the rest parts of \( o_{tar} \) that have not been learned with \( o_{src} \).

V. A CASE STUDY: TRANSFER LEARNING TO OPEN LIDS

We used the humanoid robot NAO in an actual household task for garbage can manipulation. We focus on transfer learning of affordances in order to open different types of lids. We assume that all garbage can parts are reachable and manipulable by NAO. Videos are available at: https://www.youtube.com/user/cwang1985.

A. Task Setting

In our experiment, we used five garbage cans to test our affordance learning and transfer model (see Fig. 1). The set of objects was denoted by \( O = \{o_1, o_2, o_3, o_4, o_5\} \). They were presented to NAO in sequence, and NAO decided by itself whether to transfer the learned affordances or not. \( o_1 \) and \( o_3 \) had pushable lids (Fig. 1(a) and Fig. 1(c)), \( o_2 \) and \( o_4 \) had liftable lids (Fig. 1(b) and Fig. 1(d)) and \( o_5 \) had a pullable lid (Fig. 1(e)). In each learning trial, a garbage can was positioned approximately 10 to 12 cm in front of NAO and the area to be explored was about 25 to 45 cm high. These values agreed with the capabilities of NAO due to its height and the length of its arms. The left arm of NAO was used to interact with the garbage cans.

The bottom camera on NAO’s head was used as the main sensory input (640 \( \times \) 480 resolution). For each garbage can,
a blue marker (5 cm × 2 cm) was used for the recognition of lid (with a NAO marker at its center), and a red marker (5 cm × 2 cm) for the recognition of the garbage can body, and a green marker for the recognition of the handle (10 cm × 3 cm × 1 cm), if there was one. The camera images were transformed into HSV color space. Color blobs and the bounding boxes were obtained using the OpenCV library. As a result, object parts were denoted by \( \Psi_{ai} = \{ \psi_{1i}, \psi_{bi} \}, i = 1, 3, \) and \( \Psi_{o} = \{ \psi_{o}, \psi_{l}, \psi_{h} \} \) where \( \psi_{l} \) denoted a lid, \( \psi_{h} \) denoted a handle, and \( \psi_{b} \) denoted a body. For each \( \psi \in \Psi_{o} \), its state \( s_{\psi} = (x_{\psi}, y_{\psi}) \) was obtained as the 2D coordinates of the bounding box center. Then, the spatial relation between two parts is obtained by comparing \( y_{\psi} \).

The set of action types for learning affordances was \( A = \{ a_{1}, a_{2}, a_{3}, a_{4}, a_{5}, a_{6} \} \), which were \{ sweep left, sweep right, push forward, pull back, lift up, sweep down \}. Each action type \( a_{i} \in A \) was constrained by the action parameter \( \theta \) in a subset of action parameter space \( \Theta \subset \mathbb{R}^{3} \). For example, in the Cartesian space of NAO, a push action was defined in the parameter space \( \Theta = \{ \mathbb{R}^{3} | 0 < x < 0.15, y = 0, z = 0 \} \) (in meters), and the corresponding exploration parameter space was \( \Theta_{exp} = \{ \mathbb{R}^{3} | 0 < x < 0.05, y = 0, z = 0 \} \).

To detect the opened area, we put a black plastic bag in each garbage can and calculated the area of the dark part in an captured image. The opened area was also located by a bounding box with a size of \( w \times h \) in pixels. Then, \( s_{o} \) is the absolute value of opened width in meters:

\[
\begin{align*}
  s_{o} &= \alpha h \\
  \text{where } \alpha \text{ was used to normalize } h \text{ according to the relative size of known markers. In all experiments, initial object states were the same } s_{o}^{l} = 0 \text{ when the lids were tightly closed, and the goal states were also the same } s_{o}^{g} = 0.1 \text{ m.}
\end{align*}
\]

After an action was performed on \( \psi \), the new states \( \psi' = (x'_{\psi}, y'_{\psi}) \) and \( s'_{o} \) were extracted from a new image. The displacement of \( e_{\psi} \) was calculated from (2)

\[
\begin{align*}
  e_{\psi} &= \alpha \sqrt{(x'_{\psi} - x_{\psi})^{2} + (y'_{\psi} - y_{\psi})^{2}}
\end{align*}
\]

For this learning problem, we chose a linear regression model to approximate \( g_{a} \) and \( h_{a} \):

\[
\begin{align*}
  \hat{g}_{a}(a, \theta) &= b_{j, k} + c_{j, k} \theta \\
  \hat{h}_{a}(e_{\psi}) &= d_{j, k} + e_{j, k} e_{\psi}
\end{align*}
\]

The reason for this choice was due to the character of the functional parts operation, i.e. pushing or lifting further results in more opening.

The parameters \( b_{j, k} \) and \( c_{j, k} \) in (17) were estimated by minimizing the residual sum of squares (RSS) using \( m \) observation samples:

\[
\begin{align*}
  S_{m} &= \sum_{i=1}^{m} (e_{i} - \hat{g}_{a}(a_{k}, \theta_{i}))^{2}
\end{align*}
\]

We obtained the parameters of (18) in the same way. We set \( n = 5 \) in Algorithm 1 and 3. The regressions (17) and (18) were considered accurate when the following condition was met:

\[
\begin{align*}
  |S_{m+1} - S_{m}| < \delta_{3}
\end{align*}
\]

where \( \delta_{3} \) was a small positive real value.

**B. Results**

As a baseline, we first ran Algorithm 1 for five garbage cans without transfer learning. Then, we used the learned affordances to evaluate the transfer results. In total, 156 regression models were learned as (17) and (18). Some of the results are shown in Table II.

**TABLE II**

**LEARNED AFORDANCES WITH THE FIVE OBJECTS.**

\[
\begin{array}{|c|c|c|c|}
\hline
\sigma & \psi & \alpha & \delta_{a}(a, \theta) \\hline
\hline
o_{1} & \psi_{1} & \text{push} & -0.004 + 0.574 \theta \\hline
o_{2} & \psi_{2} & \text{push} & 0.588 \theta \\hline
o_{3} & \psi_{3} & \text{pull} & -0.001 + 0.452 \theta \\hline
o_{4} & \psi_{4} & \text{lift} & 0.569 \theta \\hline
o_{5} & \psi_{5} & \text{lift} & -0.001 + 0.642 \theta \\hline
o_{6} & \psi_{6} & \text{pull} & 0.001 + 0.609 \theta \\hline
o_{7} & \psi_{7} & \text{sweep} & 0.001 + 2.891 \theta \\hline
o_{8} & \psi_{8} & \text{sweep} & -0.009 + 2.205 \theta \\hline
o_{9} & \psi_{9} & \text{push} & 0.001 + 0.240 \theta \\hline
o_{10} & \psi_{10} & \text{lift} & 0.522 \theta \\hline
o_{11} & \psi_{11} & \text{push} & 0.006 + 1.070 \theta \\hline
\hline
\end{array}
\]

The five functional parts are marked in bold, they were all correctly selected using the learned models. Also, correct action types and associated parameters were selected to satisfy (11). Due to sensory noise and the design of garbage cans, some pairs of part and action type also resulted in the observation of opening, e.g., \( (\psi_{h1}, \text{pull}) \) and \( (\psi_{h1}, \text{push}) \). However, their estimation of action parameter was out of the allowed action parameter range or the predicted opening could not satisfy the goal condition.

Table III illustrates the number of required exploratory actions with affordance transfer \( (N_{tran}) \) and without transfer \( (N_{tar}) \), together with the chosen source objects \( (o_{src}) \). If there was one.

1) Without knowledge transfer: NAO typically interacted with every object part with all 6 action types. If an action type \( a \) resulted in no significant displacement of \( \psi \), (20) would be satisfied to terminate this action. For example, sweeping left over a pushable lid for 3 times was enough to learn an accurate prediction that this action was not effective for this type of lid. It required 36 exploratory actions, which

**TABLE III**

**COMPARISON OF THE REQUIRED NUMBER OF ACTIONS FOR LEARNING**

**TARGET OBJECT \( o_{tar} \) WITHOUT TRANSFER \( (N_{tar}) \) AND WITH TRANSFER \( (N_{tran}) \) FROM SOURCE OBJECT \( o_{src} \).**

\[
\begin{array}{|c|c|c|c|}
\hline
\sigma_{tar} & N_{tar} & N_{tran} & o_{src} \\hline
\hline
o_{1} & 36 & - & - \\hline
o_{2} & 65 & 29 & o_{1} \\hline
o_{3} & 53 & 1 & o_{1} \\hline
o_{4} & 83 & 3 & o_{2} \\hline
o_{5} & 80 & 11 & o_{2} \\hline
\hline
\end{array}
\]

\[1\text{http://opencv.org/} \]
was less than executing each action type with \( n = 5 \) parameters, i.e., \( 2 \times 6 \times 5 = 60 \) samples for \( o_1 \).

2) With knowledge transfer: The most efficient transfer learning happened for learning \( o_3 \) and \( o_4 \) after \( o_1 \) and \( o_2 \) had been learned. Because \( o_1 \) and \( o_3 \) had the same pushable lid which was above the body part, while \( o_2 \) and \( o_4 \) had the same liftable handle as well as the same spatial relations. In the case of learning \( o_3 \), object \( o_1 \) was selected as the source object because \( o_1 \) was more similar with \( o_3 \) than \( o_2 \). The lid (\( \psi_{h_3} \)) and the “push” action (\( a_3 \)) were selected, with a parameter 0.087 applied, which resulted in the state of opening \( s'_{o_3} = 0.14 > s_{o_3}^{5} \). The goal state was achieved with only one single action. In the case of learning \( o_4 \), object \( o_2 \) was selected as the learning source because it was more similar than \( o_1 \) and \( o_3 \). NAO tried the lift action (\( a_5 \)) on the handle (\( \psi_{h_4} \)) to achieve the desired opening, each time with a bigger lift parameter given by Line 9 in Algorithm 3, where \( \nabla \hat{g}_{o_2}(a_5, \theta) = 1.866 \) (see Table II). This parameter selection policy suggested that lifting \( \psi_{h_3} \) more would result in more opening.

However, lifting the handle did not work for \( o_5 \) when \( o_2 \) was selected as the learning source. Then, NAO tried action types in \( a \in A \setminus \{ a_3 \} \) (Lines 13 through 19 in Algorithm 3). It happened that for pulling (\( a_4 \)) the handle generated a different effect than model \( \hat{g}_{o_2}(a_4, \theta) \) predicted. Therefore, it tried several more pulling parameters to learn an accurate prediction model \( \hat{g}_{o_4}(a_4, \theta) \). The goal condition was satisfied in Line 17 and the learning was terminated.

The most difficult case was learning \( o_2 \) when \( o_1 \) was selected as the source object. NAO started with pushing the lid, which was found ineffective. Thereafter, other action types were tried on the lid to check whether they were still ineffective as anticipated for \( \psi_{l_1} \in \Psi_{o_1} \). Similarly, the robot checked the learned body part of \( o_1 \) with a small number of actions (Lines 20 through 24 in Algorithm 3). Finally, the robot interacted with the handle which was not learned before. In total, it took 29 exploratory actions, still fewer than 61 without transfer. Because NAO reused the learned models with \( o_1 \) for \( o_2 \), rather than learned models from scratch for all (\( \psi, a \) pairs.

VI. CONCLUSIONS

In this paper, we investigated an approach for learning and transferring affordances of household products composed of several parts. Affordances were learned on-line by function approximators and used for action selection during task execution. We have demonstrated that the robot is able to acquire accurate affordance models and speed up the lid opening task by transfer learning of affordances. In the future, we will investigate more complex objects which have more than one functional part and a variety of spatial relations between parts. Also, action parameter learning in high dimensional spaces will be considered by using sophisticated non-linear regression methods.

REFERENCES