Multi-step planning with learned effects of (possibly partial) action executions

Utku Bozdogan\textsuperscript{a} and Emre Ugur\textsuperscript{b}

\textsuperscript{a,b}Department of Computer Engineering, Bogazici University, Istanbul, Turkiye

ARTICLE HISTORY
Compiled March 17, 2023

ABSTRACT

In this paper, we propose an affordance model, which is built on Conditional Neural Processes, that can predict effect trajectories given objects, action or effect information at any time. Affordances are represented in a latent representation that combines object, action and effect channels. This model allows us to make predictions of intermediate effects expected to be obtained from partial action executions, and this capability is used to make multi-step plans that include partial actions in order to achieve goals. We first show that our model can make accurate continuous effect predictions. We compared our model with a recent LSTM-based effect predictor using an existing dataset that includes lever-up actions. Next, we showed that our model can generate accurate effect predictions for push and grasp actions. Finally, we showed that our system can generate successful multi-step plans in order to bring objects to desired positions. Importantly, the proposed system generated more accurate and effective plans with partial action executions compared to plans that only consider full action executions. Although continuous effect prediction and multi-step planning based on learning affordances have been

CONTACT Utku Bozdogan Email: utku-bozdogan@hotmail.com
studied in the literature, continuous affordance and effect predictions have not been utilized in making accurate and fine-grained plans.

KEYWORDS
Affordances, effect prediction, object motion trajectory prediction, multi-step planning

1. Introduction

From a robotics standpoint, learning to predict the effects of a robot’s actions beforehand would be a beneficial skill for robots, since this can prevent potential failures and dangerous situations to the robot and those around it, and enable planning for achieving certain goals. Planning for multi-step tasks in the real world is difficult, and a generalized approach to solving this problem is even more so. Due to its difficulty, previous works compromise on certain aspects, such as predefining effect categories or object categories, simplifying the task. Discretizing information in the continuous sensorimotor space with the purpose of high-level symbolic planning with symbols may result in inaccurate plans.

In robotics, learned affordances have been used to choose objects for manipulation, to make discrete and continuous effect predictions given objects and actions, to make plans to achieve goals [1–4]. However, to the best of our knowledge, there is no single framework that can predict effects given objects and action, predict required actions to achieve desired goals, can predict the movement trajectory of the objects in response to parametric actions and can make plans composed of sequence of actions, including partial action executions, in order to achieve given goals. The contribution of our work is as follows:

- A novel latent representation for affordances: Considering affordances as relations between objects, actions and effects, our neural network based architecture forms a representation that encodes object, continuous action and continuous effect information in a single latent layer.
- Accurate continuous effect prediction: Our system can predict the motion trajectories of the objects expected to be generated by parametric robot actions. We showed that the high prediction performance compared to a strong baseline that also make affordance-based continuous effect prediction.
- Multi-step planning with partial actions: We exploited the outcome prediction capability of our system for partial actions in order to form multi-step plans that may include full or partial action executions.

2. Related work

Early affordances work on effect predictions In early works such as [5], objects were required to be recognized first, and object-specific affordances were learned from robot’s interaction experience with those objects, therefore the learned affordances could not be generalized to different/novel objects. [6] studies learning of traversability affordance where the LIDAR input was directly processed without any object detection step. Therefore, ‘direct perception’ aspect of affordance perception was realized in that work, however only a single pre-defined affordance category was addressed. Effect categories and affordances were discovered by the robot in [7,8] via unsupervised clustering techniques. However, unsupervised clustering results depend on the
particular clustering method and the feature space that might be composed of values with different metrics such as distance, frequency, angle, etc. In [9], this is taken one step further and hierarchical clustering is made over channels for better effect category discovery. Both [8] and [9] also enable forward chaining for multi-step planning. [10] propagated affordance predictions by exploiting similarities among object properties, action parameters and resulting effects using Maximum Margin Multi-Valued Regression (MMMVR), obtaining efficient affordance learning; however affordance categories were also pre-defined in that study. In [11], complex affordance learning was bootstrapped through using pre-learned basic-affordances as additional inputs of the complex affordance predictors or as cues in selecting the next objects to explore during learning. In [12] the objects were represented via point clouds, in a non-parametric manner, to provide direct perception and the grasping and a single pouring action were shown to generalize well on novel objects but result in single-step plan only. Bayesian Networks were used [13,14], enabling bidirectional predictive capabilities using the robot’s own interaction experience, but clustering is performed on object features and effects in a predefined manner. In this paper, we also study acquiring bi-directional prediction capabilities. Different from previous work, we do not find effect clusters or do not only aim predicting the final effect. Instead, our system aims to learn predicting the complete effect/action trajectory during action execution.

**Learning affordances for planning** [15] learned affordances for symbolic planning, by again clustering effect categories and using object categories as a collection of effect categories obtainable by actions available to the robot. This enables the representation of nonlinear relations in planning, however, their discrete representation while making planning easy, makes the estimations approximate, increasing long-horizon planning errors. This work was validated on a real-world setup with simple actions such as poke, grasp, and release, and also with a more complex action which is stack. The experience of the robot enabled it to gain experience from the simple actions and after experiencing the stack action enabled it to generate a valid plan for the stacking action. This framework was extended in [16] by enabling the robot to progressively update the previously learned concepts and rules in order to better deal with novel situations that appear during multi-step action executions. Similar planning capabilities were obtained by the robot using deep encoder-decoder neural networks with binary bottleneck layers [17] and with multi-head attention mechanism [18] by directly processing the pixel images instead of using hand-coded features. In [19], probabilistic planning with symbols using parameterized actions was applied to a real robot task, showing that continuous tasks can be performed with discrete planners using parameterized behaviours. Different from the previous work where only final outcomes were used for planning, our system can exploit intermediate effects expected to be observed for example in the middle of the action execution for planning.

**Learning visual grasp affordances** Using RGB/RGBD images for predicting affordance classes or pixel-wise affordance labels for object manipulation has become popular in the recent years [20–25] and was shown to be a feasible approach for learning how to grasp different objects. Similarly, [26] are able to learn the affordances of objects such that they can place objects/humans in correct poses in a scene and also choose the correct object type to place in a given scene. [27] uses point clouds to learn general geometric features from object interactions, enabling them to place objects in a scene correctly. In [28], a generative model learned from interaction images was
used to propose potential affordances. The aim was to learn a generalizable prior from interaction data and then utilize it to propose reasonable goals for unseen objects. These goals were then attempted to be executed by an offline RL policy, learned from interaction data, and tuned online efficiently to adapt to unseen objects. However, continuous effect prediction and multi-step planning aspects were not addressed in these studies.

**Affordances for efficient learning** Affordances can also be used to reduce the search space in order to efficiently generate plans to solve long-horizon tasks. Recent approaches utilizing this idea extended the definition of affordances to represent not only single-step action-effect knowledge but as action feasibilities [29], or intents which are similar to goals [30] in order to make affordances useful for multi-step plans. However, the feasibility concept of [29] accepts a grasp action achieving nothing as afforded, and would also accept a grasp action as afforded regardless of whether it was an appropriate grasp for an object. While [30] overcame this with intent representation, intents were specified a priori in their work and although a sub-goal discovery direction was proposed for learning, it was not explored.

**Learning continuous effect trajectories** Similar to our work, [31] and [32] learned to predict the full motion trajectory of an object, using the robot’s own interaction experience, and top-down images of the objects. These studies are important for their inclusion of the temporal aspect of effects. The utility of different features was also investigated, such as hand-crafted shape features, CNN extracted features, or support point features extracted from a neural network. The authors used these features and the interaction experience to train recurrent neural networks and were able to accurately predict trajectories resulting from a lever-up action in a real-world setting with multiple objects. A common shortcoming of the aforementioned methods which make predictions for action or effects is related to the use of recurrent methods for long-horizon tasks. The use of recurrent networks such as LSTMs [33] or GRUs [34] is shown to be effective for short-term predictions. However, their recurrent structure causes any error in their prediction to accumulate over time, causing lower success rates in executing long-horizon tasks [35]. In this paper, we compare the effect prediction capabilities of recurrent neural network based systems and our conditional neural process based system.

### 3. Proposed Method

#### 3.1. General Architecture

We propose a framework that can learn (i) predicting the effect trajectory given initial object image and action execution trajectory and (ii) finding the required action execution trajectory to achieve a desired effect on a given object. Our system is built on top of Conditional Neural Processes (CNPs) [36] that bring together the inference potential of Gaussian Processes and the training of neural networks with gradient descent by learning a prior from data. As we would like to predict both actions and effect from objects and (possibly missing) actions and effects, we propose a neural network structure that takes the object image as input together with action and/or effect values at different time points. In other words, given initial object image, our network can be conditioned with action and/or effect values at different time points in
order to generate action and effect values for all time points during action execution. The general structure of the proposed system is shown in Fig. 1. As shown the system is composed of two encoders that process action and effect information at different time points, an image encoder that processes the depth image of the object, averaging operations to acquire a common object-action-effect representation (r), which in turn can be used to predict action and effect values at other time points using the corresponding two decoders.

In detail, inspired from Deep Modality Blending Networks [37], our system encodes information coming from different channels (object, action and effect) into a common latent space, using a weighted average of encoded modality representations, facilitating information sharing, and also providing a regularization effect on the representations learned, similar to dropout [38]. Each channel is encoded separately by its own encoder, and the latent representations are subjected to a weighted averaging operation:

$$r_i = \sum h_a(a_{t \text{obj}_k}) * w_1 + h_e(e_{t \text{obj}_k}) * w_2$$  \hspace{1cm} (1)

where $w_1 + w_2 = 1$, and

$$r = r_1 \oplus r_2 \oplus r_3 \oplus ... \oplus r_i$$  \hspace{1cm} (2)

where the commutative operation used is averaging. Then, depth image features and target time-step are concatenated to form the merged representation $r_{\text{merge}} = \{r, f(\gamma_0), t_{\text{target}}\}$.

This merged representation is then decoded by separate decoders, each corresponding to a different channel. This merged representation is decoded at the action decoder.
by

\[ g_a(r_{mrg}) = (\mu_{a_{target}}, \sigma_{a_{target}}), \] (3)

and the effect decoder by

\[ g_e(r_{mrg}) = (\mu_{e_{target}}, \sigma_{e_{target}}), \] (4)

to yield predictions for action and/or effect for the target time step shown in Fig 1.

The latent representation \( r \) can be viewed as the shared affordance representation for actions \( a \) and effects \( e \) for different objects \( o \). Learned affordances can then be used to predict the effects of actions or the required action to generate target effects. This model can be used to create multi-step action plans to achieve goals beyond single action executions by chaining the predictions.

### 3.2. Training

In our implementation, the system is object-centric. An action is defined in terms of the distance of the robot’s end-effector to the object. An effect is defined as the displacement of the object from its starting position throughout an action. A depth image of the object is included as an external parameter \( \gamma \) for the action. \( t_i \) is the \( i \)th time-step of an interaction trajectory from the dataset \( D \).

\[ D_d = \{(a_t, e_t, o, t)_{t=1}^{t=1}\}_{d=1}^{d=m} \] (5)

is an interaction trajectory where \( 1 \leq d \leq m \) and \( m \) is the number of trajectories in the data set \( D \). \( 0 \leq t \leq 1 \) is a phase variable in control of the passage of time where \( t \in \mathbb{R} \).

At each training iteration, \( k \) observations are sampled uniformly at random from a randomly selected interaction trajectory \( D_d \) where \( 1 \leq k \leq obs_{max} \), \( k \in \mathbb{N} \) is also sampled uniformly at random and \( obs_{max} \) is a hyper-parameter denoting the maximum number of observations that the model is allowed to use during one iteration. These observations are then encoded and aggregated. A cropped object depth image is encoded separately on a CNN encoder network, and the resulting vector is concatenated at the end of this aggregated representation. Before a prediction can be made, a target time-step is also concatenated after the image features. Finally, this merged representation is decoded to yield predictions for action and/or effect for the target time step shown in Fig 1. Gradient descent is used with the loss function (6) with Adam optimizer [39].

\[ \mathcal{L}(\theta, \delta) = -\log P(y_j | \mu_j, softmax(\sigma_j)) \] (6)

After training, the network is able to predict the entire interaction trajectory given a single observation at \( t = 0 \). An A* planner [40] is then used on top of the network to solve tasks requiring multiple actions and steps.
3.3. Actions

Our arm-hand robot is equipped with two actions, namely push and grasp. The parametric push action is specified by an angle $\theta \in [0, 2\pi]$; a push distance $l = 0.05$ and a radius $= 0.2$ both in meters. The gripper starts the push execution from the red circumference of a circle of radius (radius = 20cm) centered around the object (see Figure 2), at an angle $\theta$ and pushes the object $l$ meters from its center of mass.

Larger sized objects may be displaced more as a result of this setup. The model is expected to learn the rollability and pushability affordances from these interactions and based on the object shape be able to predict the trajectories of rollable objects such as spheres or lying cylinders and non-rollable objects such as cuboids or upright cylinders.

Grasp actions, on the other hand, are realized by lowering the open gripper to grasp position, attempting to grasp an object by closing the gripper and lifting the gripper up. Our model is expected to learn the graspability affordance from these interactions. Based on object size and shape, it should also be able to predict the interaction trajectories.

3.4. Planning with partial action executions

The A* planner (with Euclidean distance to the goal heuristic) is used to generate a sequence of actions to move the object from its initial position to a goal position. Each branch in the search tree corresponds to either a push action or a grasp action. As grasp action is not parameterized, there is a single branch for grasp action. On the other hand, as a push action might be applied from different approach angles and for different push distances, the range of possible approach angles and push distances are discretized and used to create multiple branches from the same node in the search tree. The initial 20%, 40%, 60%, 80% and 100% segments of push action was considered during search. Additionally, the push direction was discretized into 36 directions. Therefore, the branching factor was set to 180.
The planner uses predictions of actions with different parameters to update the predicted location of the object. The search is completed if the difference between the predicted and goal object position is less than 2 cm. Our model is able to work with continuous inputs, however, the planner can only propose a finite amount of actions due to the mentioned discretization design choice.

Interactions with single and multiple actions are generated, which also include partial actions. Partial actions are when an action is started to be applied, however it is not completed, i.e. it is executed partially. For example in a push action, the push may be cut short before the gripper even contacts the object, or the gripper reaches the center of mass of the object, meaning that the object has already started being pushed, but the push is not completed yet. Importantly, our model is trained only with full action interactions. Yet, our model can generate the effect at any desired time point, and therefore it can predict the consequence of partial action executions. Planner can use effects of such partial action executions to generate plans with finer resolutions (compared to plans that can only include full action executions).

Note that an action can be applied only if the object is reachable by the robot. Our model is expected to learn the reachability affordance from its interaction experience and actions are only considered during planning if the object is reachable.

4. Experiments results

4.1. Effect prediction performance

In this paper, we propose a novel system to predict effects given objects and actions. Furthermore, our system can generate complete motion trajectories of objects as effects rather than their final positions. In order to assess the performance of our system, we compare our system with a recent study that can also predict motion trajectories of objects using CNN and Long short Term Memory (LSTM) model [31]. We used the same dataset that [31] used where lever-up actions were applied to objects with different geometric shapes from different points. An example lever-up action is shown in Figure 3. The objects had different number of edges (between 3 and 8), sizes and orientations. The objects were levered-up from different contact points. The authors translated and rotated the top-down 128x128 grayscale image according to the contact point and lever-up direction in order to simplify the prediction problem. The dataset was separated randomly into 80% training, 10% validation, and 10% test as in [31].

The experiments were performed with 5-fold cross-validation and early stopping with one million iterations. For comparison, we gathered n-step predictions, taking 15 previous steps as observations, and compared the results with the ones reported in the same manner in [31]. As shown in Figure 4, the error between the predicted and actual positions of the objects in [31] varies between 0.90 – 1.00cm, whereas the error
Table 1. N-step prediction errors obtained in our model.

<table>
<thead>
<tr>
<th>Step</th>
<th>Error (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-step</td>
<td>0.340 ± 0.044</td>
</tr>
<tr>
<td>2-step</td>
<td>0.352 ± 0.046</td>
</tr>
<tr>
<td>3-step</td>
<td>0.364 ± 0.049</td>
</tr>
<tr>
<td>4-step</td>
<td>0.375 ± 0.051</td>
</tr>
<tr>
<td>5-step</td>
<td>0.387 ± 0.054</td>
</tr>
</tbody>
</table>

Figure 4. N-step error plots obtained in the LSTM approach, reused with permission [31].

was in the range of 0.3 – 0.4cm in our system (Table 1). This comparison shows our model yields significantly lower error rates compared to a recurrent method, since the output is predicted directly avoiding error accumulation for multi-step predictions.

4.2. Training Environment

A simulated scene was constructed in CoppeliaSim [41] as shown in Figure 2. A UR10 robot interacts with objects of different shapes and sizes by applying push and grasp actions on a tabletop setting. A Kinect sensor is placed above the table vertically such that the entire table is visible. The parts of the interaction where the object is potentially going to be displaced are recorded. The recorded information consists of action and effect data from once every 3 simulation steps (a single step is 50ms), which is chosen empirically, and a single depth image of the table with the object on top taken at the beginning of each interaction. The simulation dataset is split into training (%80), validation (%10) and test (%10) sets.

4.3. Single-action push and grasp effect prediction

Different models were trained for push and grasp actions separately. The simulation data sets were always split between 80% training, 10% validation, and 10% test data. For all the results reported in this work 10-fold cross-validation was applied unless otherwise specified. The models were trained for one million iterations, without batches due to variable length of inputs and early stopping was employed. The learning rate was set to 1e–4. All errors reported in meters denote the distances to a specified goal position. The predictions for a single action is fixed to take 25 time steps.

For the push action, a data set made up of 500 trajectories was used. For each interaction, objects were chosen randomly and placed at the center of the table. An angle for the push $\theta \in [0, 2\pi]$ was chosen randomly. The robot performed a complete push action and the resulting interaction data was recorded. The error in predicting the final position of the object was found to be around 0.02m as shown in Figure 6.
Table 2. Single grasp action prediction results on variable sized objects placed on a fixed location.

<table>
<thead>
<tr>
<th>Error (m)</th>
<th>True Positive (%)</th>
<th>True Negative (%)</th>
<th>False Positive (%)</th>
<th>False Negative (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.275 ± 0.044</td>
<td>75.17</td>
<td>90.48</td>
<td>9.52</td>
<td>24.82</td>
</tr>
</tbody>
</table>

Similar to our analysis in the previous subsection, our system was shown to be effective in predicting object motion trajectories in push-like actions.

For the grasp action, a data set made up of 100 trajectories was used. Objects of varying sizes were randomly chosen to be placed at the center of the table. The robot then performed the grasp action and the resulting interaction data was recorded. The results for the grasp action are provided in Table 2. In interpreting the performance and success of graspability prediction, if the change in height is larger than 0.1 meters, the grasp was assumed to be a success. If the change in height is less than 0.1 meters, it is a failed grasp. If the test data does not have a significant change in its z-axis coordinates but the predictions do, then this is a false positive and finally, if the test data has a significant change in z-axis coordinates but the predictions do not, then it is a false negative. We have found that the robot had more difficulty grasping non-rollable objects of equal size, most likely due to the fact that a cube and an upright cylinder are both grasped by straight surfaces whereas large spheres and sideways cylinders are grasped by curved surfaces from points that are located above their center of mass, causing them to slip easier.

As shown, our system could successfully predict graspability affordance. However, it is important to note that the average grasp action error is significantly larger than the average push action error. The mean error was relatively high because the incorrect graspability predictions generated high positional errors that significantly increase the mean error value. This is potentially due to unsuccessful grasps. In the event of an unsuccessful grasp, the object may slowly slip from the robot’s hands and land on the table close to its initial position. The object may topple or roll (sometimes off the table), leading to position changes that are uncertain beforehand and therefore cannot be accurately predicted.

4.4. Planning performance

Next, our model is requested to generate plans to bring the object to goal positions that might be beyond the range of single pushes or closer than a full-push. Therefore, the planner is expected to generate sequences of actions that might include partial executions as well. A sample plan execution is shown in Figure 5, where the plan is composed of two actions. The goal is shown with a blue box, the actual object positions are shown with red color and the predictions are shown with green color.

Given goal positions, we run our model in three modes:

- predict one push action to reach a goal, which is maximum one-step ahead,
- predict one (possibly partial) push action to reach a goal, which is maximum one-step ahead,
- predict sequence of (possibly partial) push actions to reach a goal, which is maximum three-steps ahead.

After plans are made, they are executed by the robot. The distance between the goal position and final actual position of the object is reported as error. The results are provided in Figure 6. As shown, the object can be brought to the goal position more
Figure 5. Results of applying the model’s predicted actions in a scene. Images are ordered from left to right. The top row is from the first action execution, the bottom row is from the second action execution. The blue object denotes the target position, the red object is generated from the robot’s effect predictions. The blue and red objects are not interactable by the robot. The green object is interactable and is acted upon by the robot, to show the ground truth results of the robot’s predicted actions.

accurately if the partial actions are considered during planning even in single action execution case. Even error obtained in multiple (potentially partial) action executions is smaller than single full action execution. This shows the effectiveness of our system in making accurate plans. For the results reported in Figure 6, the single full setting is tested with 50 goals, the single partial setting with 100 goals, and the multi partial setting with 80 goals.

5. Conclusion

In this paper, we realized a model for multi-step action and effect prediction. While previous work’s utilization of bidirectional learning is limited, our model specifically creates its latent representations using this concept and is able to make multi-step predictions that are in accordance with ground truth manipulations. We emphasize using object-centric inputs to achieve generalizability and investigate simple affordances of several classes of objects of different sizes. By using a network for single interaction predictions which can be interpreted similar to a state transition function and pairing it with a planner with heuristics to propose goal-directed actions the model was shown to achieve low error in reaching target positions. While the results of our experiments
are promising, the model still requires verification in the real world. Our next step is gathering data and testing our implementation with a real robot.

Our work uses a conditional architecture to avoid the compounding error problems of recurrent architectures and models that are used in a similar way by feeding their current step output as next step input. This advantage of using conditional models is shown in this work against using an LSTM network, and against a Multimodal Variational Autoencoder (MVAE) in [37]. Recently, transformer models gained popularity as being a good alternative to recurrent models. By using the attention mechanism they eliminate the need for recurrence, and attention can potentially be beneficial for our model as well. We plan to investigate the capabilities of such models in future.

6. Disclosure statement

No potential conflict of interest was reported by the authors.

Acknowledgement(s)

The numerical calculations reported in this paper were partially performed at TUBITAK ULAKBIM, High Performance and Grid Computing Center (TRUBA resources). Authors would like thank to Alper Ahmetoglu for providing insightful comments for this paper.

Funding

This research was supported by TUBITAK (The Scientific and Technological Research Council of Turkey) ARDEB; 1001 program (project number: 120E274); TUBITAK BIDEB; 2210-A program; and by the BAGEP Award of the Science Academy.
References


Available from: https://doi.org/10.1109/tssc.1968.300136