Affordance Blending Networks

Hakan Aktas[1], Yukie Nagai[2], Minoru Asada[3,4], Erhan Oztop[3,5], Emre Ugur[1]

***Abstract*—Affordances, a concept rooted in ecological psychology and pioneered by James J. Gibson, have emerged as a fundamental framework for understanding the dynamic relationship between individuals and their environments. Expanding beyond traditional perceptual and cognitive paradigms, affordances represent the inherent effect and action possibilities that objects offer to the agents within a given context. As a theoretical lens, affordances bridge the gap between effect and action, providing a nuanced understanding of the connections between agents' actions on entities and the effect of these actions. In this study, we propose a model that unifies object, action and effect into a single latent representation in a common latent space that is shared between all affordances that we call the affordance space. Using this affordance space, our system is able to generate effect trajectories when action and object are given and is able to generate action trajectories when effect trajectories and objects are given. In the experiments, we showed that our model does not learn the behavior of each object but it learns the affordance relations shared by the objects that we call equivalences. In addition to simulated experiments, we showed that our model can be used for direct imitation in real-world cases. We also propose affordances as a base for Cross Embodiment transfer to link the actions of different robots. Finally, we introduce selective loss as a solution that allows valid outputs to be generated for indeterministic model inputs.**

I. Introduction

Affordances are action possibilities provided by the objects in the environment to embodied agents. In general agents learn affordances through goal-free exploration and interaction with the environment and by observing the effects they create on objects with the available actions. Affordances generally encode the relations between objects, actions, and effects and can be represented by models that predict effects given to objects and actions. In this paper, we use an affordance formalization inspired by [1] to represent single and multi-agent affordances and various equivalences. We built the structure of our model based on this representation and analyzed the latent representations it forms in regard to these equivalences.

The contribution of our work is as follows

- A deep LfD model that can learn affordances and form common latent representations between action-effect-object representations based on these affordances and the equivalences between the affordances
- Cross-embodiment learning using agent equivalences between the actions of different actions using the same model
- Cross-Embodiment transfer of learned affordances
- Direct imitation in real-world between demonstrators and robots using objects' effect trajectories
- Selective Loss, a solution to mitigate the undesired consequences of indeterministic training steps and to obtain valid outputs from indeterministic model inputs

[1]Hakan Aktas and Emre Ugur are with Computer Engineering Department, Bogazici University `hakanwhitestone@gmail.com` [2] Yukie Nagai is with IRCN, the University of Tokyo [3]Minoru Asada and Erhan Oztop are with SISREC, Osaka University [4]Minoru Asada is also affiliated with the International Professional University of Technology in Osaka [5]Erhan Oztop is also affiliated with Computer Engineering Department, Ozyegin University.

II. Related Work

Learning From Demonstration (LfD) [2], also known as Imitation Learning, has been a topic studied by many in recent years [3]–[8]. Several different LfD methods have been proposed in various studies. Some were based on statistical modeling and dynamic systems [9]–[12], while others used Locally Weighted Regression to learn the environment parameters [13]–[15]. Locally Weighted Projection Regression [16], Gaussian Mixture Models [17], [18] and Hidden Markov Models [19]–[22] are also utilized to learn motion patterns from demonstrated actions.

In recent years, neural networks have also been widely used in Learning from Demonstration systems to learn movement primitives using complex data [23]–[26]. In a previous study, the Conditional Neural Movement Primitives [27] was introduced as a deep LfD architecture that can be used to learn complex motion trajectories. This method is based on Conditional Neural Processes [28] which has the ability to construct full trajectories using any given points on arbitrary time steps. Because of this capability, in this study, we used the CNMPs as a backbone system for our model.

Cross Embodiment Transfer, the previous works on cross-embodiment transfer aim to transfer or generalize collected robot motion trajectories to various robots with different embodiments [29], [30]. These studies include transfer in and in between domains such as transfer to robots with different morphologies in simulation [31], [32], transfer to different robots in real world [33], from robots in simulation to real robots [34], [35].

Several methods have been introduced to use cross-embodiment transfer to expedite the training by using information from other agents. When the policy of the source agent is known, several studies showed that Reinforcement learning can be used to fine-tune the value function [36], the visual encoder [37], or the policy [38] of the target robot. In a previous study, we also showed that cross-embodiment transfer of learned representations can be used to bootstrap Reinforcement Learning [39]. We also proposed a method that can align and transfer the actions of the robots if a common task space can be established between them [40]. In this study, we build on that model and propose that affordances can be used as a well-defined common task space between the robots.

III. Method

A. Affordance Formalization

In this paper, we will use an affordance representation similar to the one used in [1]. However, we will be using an object-centric approach. We formalize and represent the affordances as the following,

$$(effect, (object, action)) \tag{1}$$

Here, we define the object as the entity the robot/agent interacts with and the effect as the resulting change caused by the action when the action is executed on the given object. Action can be any motion the robot/agent executes. Using this representation, we also define several equivalences. When the same action is applied to different objects and the same effect is observed, we state that their affordance for that action and effect are equivalent. We formalize this equivalence as the following,

$$(effect, (\left\{ \begin{array}{c} object1 \\ object2 \end{array} \right\}, action)) \tag{2}$$

We also define the affordance of two actions as equivalent if the effects caused by that action on the given object are the same. We formalize action equivalence as the following,

$$(effect, (object, \left\{ \begin{array}{c} action1 \\ action2 \end{array} \right\}) \tag{3}$$

When multiple agents/robots are involved, the notation for the affordance representation becomes,

$$(effect, (agent, (object, action))) \tag{4}$$

In this case, the actions of two different agents can also be equivalent and can be formalized differently based on the actions. If the actions executed by the agents are the same,

$$(effect, (\left\{ \begin{array}{c} agent1 \\ agent2 \end{array} \right\}, (object, action)) \tag{5}$$

An example of this case would be,

$$(lifted, (\left\{ \begin{array}{c} Robot1 \\ Robot2 \end{array} \right\}, (cuboid, lift)) \tag{6}$$

Assuming both $Robot1$ and $Robot2$ can lift the given object ($cuboid$) with their $lift$ action. On the other hand, if the actions performed by the agents are different, the formalization becomes,

$$(effect, \left\{ \begin{array}{c} agent1 \\ agent2 \end{array} \right., (object, \left. \begin{array}{c} action1 \\ action2 \end{array} \right\}) \tag{7}$$

An example of this case would be,

$$(moved, \left\{ \begin{array}{c} Robot1 \\ Robot2 \end{array} \right., (cuboid, \left. \begin{array}{c} push \\ pull \end{array} \right\}) \tag{8}$$

Assuming $Robot1$ can move the $cuboid$ object by using a $push$ action and $Robot2$ can move it by using a $pull$ action.

B. Model

In an earlier work [40], we showed that a multi-channel CNMP model can be used to find correspondences between robots with different morphologies. In this study, we generalize that approach to object affordances and propose affordance blending networks that can be used to form discrete affordance representations from object representations and continuous effect and action trajectories.

In our system, the objects, and continuous action and effect trajectories are encoded and learned using CNMPs, and the latent representations of the CNMPs of these trajectories are blended to form a common latent space that we call affordance space. Let A be the dataset that contains the information related to one affordance, including the action trajectories of multiple agents, the resulting effect trajectory of the object entity, and the representation of the object entity. More formally,

$$A_j = (\{a_t^{R_j}, e_t, o, t\}_{t=0}^{t=1})_j \tag{9}$$

where e_t represents the resulting effect at time t, o represents the object/entity information, t represents time and $a_t^{R_j}$ represents the sensorimotor action information of R_j at time t where $R_j = [Agent_1, Agent_2, Agent_3..., Agent_k]$ represents the agents that are related to affordance A_j. j represents the sample number and $0 <= j <= m$ where m is the size of the dataset. $0 \leq t \leq 1$ is a phase variable that represents the time where $t \in \mathbf{R}$. Not all agents are related to all affordances since it is not possible to talk about grasp affordance if the agent has no means of grasping such as a generic differential drive mobile robot. Since the goal is to convert the data to latent representations and blend them to get an affordance representation, first, the action trajectories of all the agents that are related are converted to latent representations as in [40],

$$L_i^{a_{r_j}} = E^{a_{r_j}}((t_i, a_{t_i}^{r_j})|\theta^{a_{r_j}}) \quad r_j \in R_j \tag{10}$$

where $E^{a_{r_j}}$ is an action deep encoder for agent r_j with weights $\theta^{a_{r_j}}$, and $L_i^{a_{r_j}}$ is the action latent representation of agent r_j constructed using the given observation. Then these representations are averaged for each agent,

$$L^{a_{r_j}} = \frac{1}{n}\sum_i^n L_i^{a_{r_j}} \quad r_j \in R_j \tag{11}$$

where n is the number of sampled observations during this iteration. These are then combined using convex combination to obtain the common latent action representation,

$$L^a = \sum_{r_j}^{R_j} p^{r_j} L^{a_{r_j}} \quad 0 \leq p^{r_j} \leq 1, \sum p^{r_j} = 1, r_j \in R_j \tag{12}$$

Object and effect latent representations are obtained similarly to $L_i^{a_{r_j}}$,

$$L_i^e = E^e((t_i, e_{t_i})|\theta^e) \quad L^e = \frac{1}{n}\sum_i^n L_i^e \tag{13}$$

$$L_i^o = E^o((t_i, o_{t_i})|\theta^o) \quad L^o = \frac{1}{n}\sum_i^n L_i^o \tag{14}$$

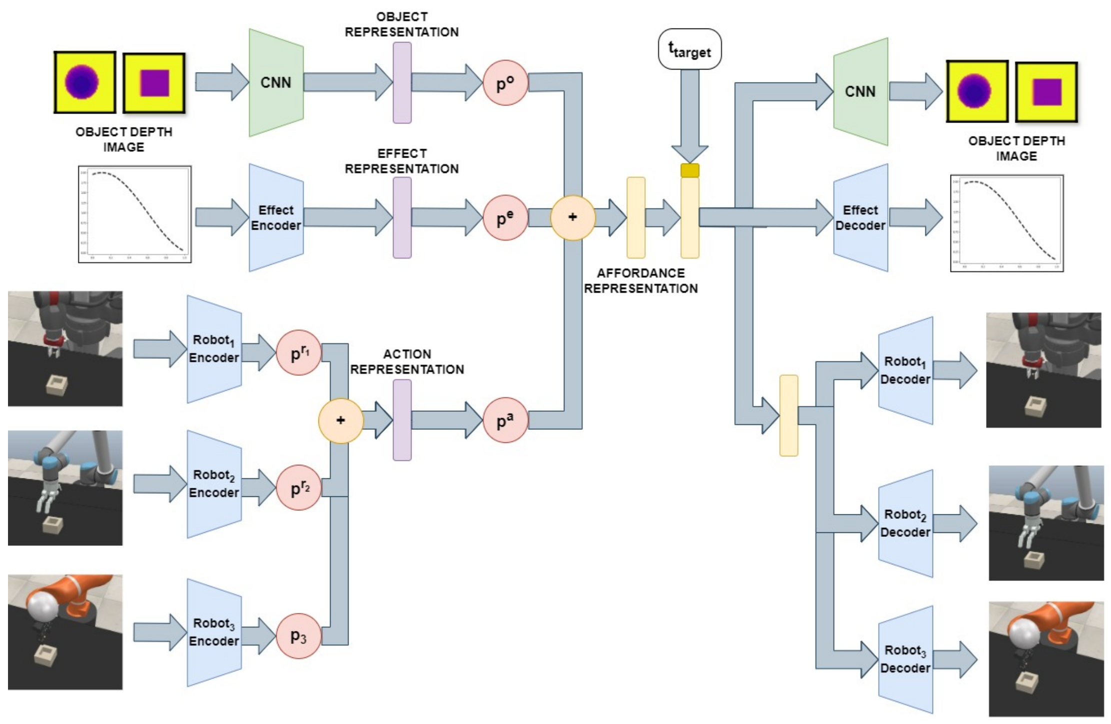

Fig. 1. The general overview of our model. Please refer to the text for the details of training and generation.

where E^e and E^o are effect and object deep encoders respectively. The obtained latent action, effect, and object representations are then combined using convex combination,

$$L^F = p^a L^a + p^e L^e + p^o L^o \quad 0 \leq p \leq 1, p^a + p^e + p^o = 1 \tag{15}$$

where L^F represents the affordance tuple that can be used to reconstruct the trajectories used to form it. After all representations are merged into one, this representation is decoded to obtain target distributions on t_{target} for the action of all agents, object and effect:

$$(\mu^{a_{r_j}}_{t_{target}}, \sigma^{a_{r_j}}_{t_{target}}) = Q^{a_{r_j}}((L^F, t_{target})|\phi^{a_{r_j}}) \quad r_j \in R_j \tag{16}$$

$$(\mu^{e}_{t_{target}}, \sigma^{e}_{t_{target}}) = Q^{e}((L^F, t_{target})|\phi^{e}) \tag{17}$$

$$(\mu^{o}_{t_{target}}, \sigma^{o}_{t_{target}}) = Q^{e}((L^F, t_{target})|\phi^{o}) \tag{18}$$

where Q^r is a deep decoder with weights ϕ^r that constructs distributions with mean $\mu^r_{t_{target}}$ and variance $\sigma^r_{t_{target}}$ for the robot r. The learning goal of our system is to obtain more accurate distributions for given condition point(s) and the target point, and therefore the loss is defined similarly to [27]:

$$\mathcal{L}(\theta, \delta) = -logP(y_j|\mu_j, softmax(\sigma_j)) \tag{19}$$

The complete model can be seen in Figure 1.

Our system can also be used to transfer affordance relations demonstrated only using one of the agents to other agents. After sufficient training with affordances known to all agents, the demonstration of the affordance known only by one agent can also be used for training. The system should continue to be trained using the old data alongside the new one to prevent Catastrophic Forgetting. After sufficient training with the new data, the system can also output the corresponding action of the other agent and can output the effect of that action when conditioned with the new object, if the affordance is shared.

C. Selective Loss

In addition, we also introduce a modification to the system's loss function, namely the negative log-likelihood function. As stated, the system uses stochastic convex combination to combine the representations of the actions, effects, and objects. This introduces a problem if more than 2 objects with similar behavior are used (such as a cuboid and upright cylinder). If the weight of the object representation is too small, the resulting affordance representation would contain too little information about the object which makes the prediction about the object in the output non-deterministic. While this does not matter in the test time since it is not possible to deterministically infer the object given the action effect couple if more than one object with similar behavior is used, during training this causes spikes in loss of the system. This occurs when the following all happen,

- the system is biased to output one of the objects over the others for the affordance that corresponds to the action-effect couple given during that training iteration
- the weight of the object representation is too small to affect the affordance representation during that training iteration
- there are multiple objects that have the behavior that corresponds to the action-effect couple given as input during that training iteration
- the desired object output during that iteration is different from the object that the system is biased to output

In this case, the system will output the biased object as output while the desired object output is different. This behavior is

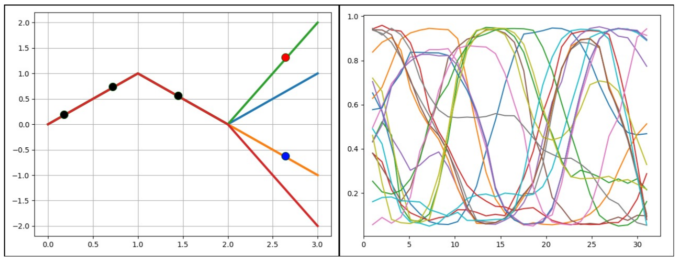

Fig. 2. An example of an indeterministic training iteration (on the right) and an example case where selective loss is a potential solution.

not desired and hurts training performance. That's why we introduce a modification to the loss of the image output layer. We call the new loss function selective loss. It is formally defined as the following,

$$\mathcal{L}_{selective} = \arg_{\mathrm{x}} \min \mathcal{L}_x(\theta, \delta) \tag{20}$$

Simply, the loss is calculated for all possible outputs and the minimum is chosen as the loss to use during that iteration. This also biases the system to output one of the objects when only action and effect representations are used to construct the affordance representation. This way, the system outputs one of the possible objects that have that affordance rather than outputting a combination of the depth images of all the objects that share that affordance which does not provide any information.

We also believe that this loss function can be used in single-channel (or vanilla) CNMP since the architecture shares the same issue when the following all happens during a training iteration,

- the sampled condition/context point combination is not novel (all shared points are shared between more than one trajectory)
- the selected target point is novel (not shared between the trajectories)
- the system is biased to another trajectory than the desired one for the sampled condition points

In this case, the system predicts the wrong trajectory, and a very high loss value is obtained. In the later stages of the training, this effect significantly increases since the variance values get smaller as training progresses. If the number of non-novel points is too many in the training set, this causes the system not to learn even when novel condition points are given as input because frequent loss spikes disrupt the learning process. An example of such training iteration can be seen in the right of Figure 2. As can seen in the figure, if not novel points are chosen as condition points (black points on the figure) and the predicted point is different from the desired point, the loss value will be significantly high.

This problem may look trivial to fix in the case of the trajectories on the left of Figure 2 since one can disclude the common parts of the trajectories from the training set, since they will not be used during testing. However, in cases like in the trajectories on the right of Figure 2 where multiple trajectories overlap and/or intersect at different intervals, this problem becomes hard to overcome with simple solutions.

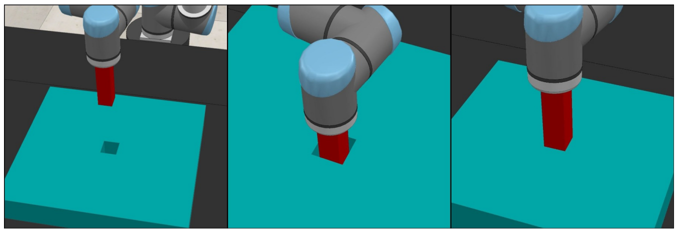

Fig. 3. The setup of the experiment conducted in Section IV-A (on the left). The resulting effect if the opening is large enough to be insertable (in the center) and the effect if the opening is not large enough (on the left).

IV. Experimental Results

A. Insertability

This experiment aims to verify that our model can encode and decode insertion affordance. A single robot (UR-10) is used in a simulated environment [41] . UR-10 is equipped with a fixed-sized red rod as can seen in Figure 3. The task is defined as inserting the rod into the opening in the planar surface on the table. The sizes of the openings vary across experiments, and if it is large enough, it is insertable by the robot (in the center of Figure 3), while it is not insertable if it is not large enough (on the right of Figure 3). The action is defined as the joint positions of the robot, and the depth image of the surface centered around the opening is used as the object. A force sensor is placed in the connector that connects the robot to the rod, and its output is used as effect input to the system. The depth images of 8 different-sized openings are used for training (4 insertable and 4 non-insertable), and 2 are used for testing (1 insertable and 1 non-insertable) to show that our model can generalize to unseen-sized objects. The sizes of the test images are chosen in between the sizes of the training images and are not chosen in between the images that are at the boundaries (where one image is inserable and the other is not) since they are not possible to predict. Some of these images can be seen in Figure 4.

The effect and action results can be seen in Figure 5. The difference between the force readings of insert and not insert conditions can be seen at the top left and center of the figure. When the opening is insertable, the force readings start to

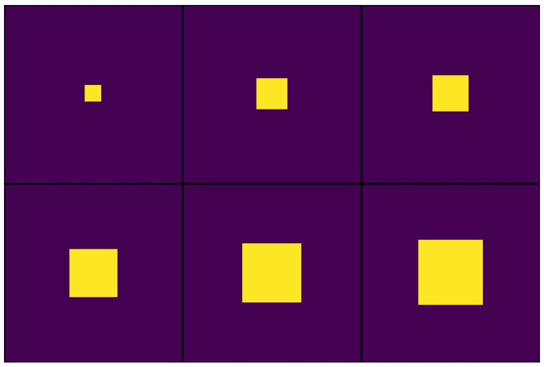

Fig. 4. The depth images of Insertable and non-insertable openings used in the experiment in Section IV-A. The images at the top are some of the insertable openings and images at the bottom are some of the non-insertable openings.

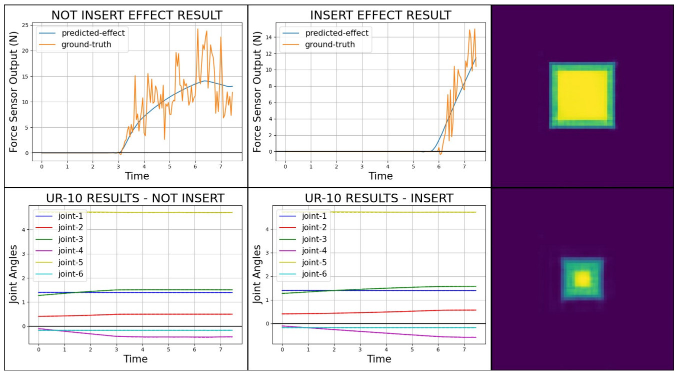

Fig. 5. The results of the insertion experiment conducted in Section IV-A. The difference between the generated and actual effect trajectories can be seen at the top of the figure where orange lines represent the ground truth and blue lines represent the predicted trajectories. The difference between the generated and actual action trajectories can be seen at the bottom where the dashed lines represent the ground truth values and the solid lines represent the model predictions. The dashed lines are hardly visible due to low error. The depth image at the top right represents the output image when the input configuration is insertable, and the bottom is the output when the input configuration is not insertable.

rise at the end of the trajectory because the tip of the rod touches the table, while when the opening is not insertable, the force readings start to rise in the middle of the trajectory because the tip of the rod touches the edges of the opening. Due to this difference in motion, the action trajectories are also slightly different, as can seen at the bottom left and center of the Figure. These prediction results can be obtained by setting any two of the three weights of the system (action, effect, and object) to 0.5. There are no weights for the robots since only one robot is used. Furthermore, if the given input configuration (combination of inputs) is insertable the system outputs the depth image at the top right of the Figure and outputs the image at the bottom left of the Figure if the input configuration is not insertable regardless of the input image given to the system (note that if the input configuration is insertable the given image has to insertable and if the configuration is not insertable the image also has to be not insertable). This shows that instead of learning different representations for each input image, our system learns a common representation that depends on the affordance.

This formation can be seen in Figure 6. The plot shows how the affordance representation changes as training progresses. To construct the affordance representations, we set the weights of the system (action, effect, and object) to 0.33. We picked several novel points along the effect and action trajectories as input to the effect and action encoders. We used all ten objects used in training and testing which are shown in the figure by different markers. Principal Component Analysis (PCA) is used to decrease the dimension of each affordance representation to two. The colorbar shows the training progress. As can seen in the figure, at the beginning of the training (on the right of the plot), the representations are closer together (due to initialization), and as the training progresses, they diverge, and two groups converge into two different points. It is hard to see all markers due to overlaps. This shows that rather than learning different affordance representations for

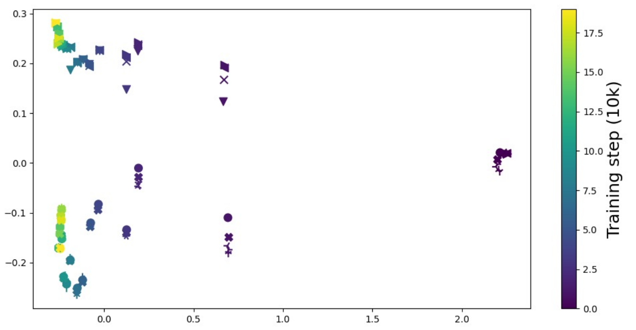

Fig. 6. Latent space analysis of the affordance representations of the experiment in Section IV-A. The plot shows how the representations constructed using different object depth images change as the training progresses. Each different marker represents a latent representation created using a different object. It can be seen from the left of the figure that the latent representations of objects that share the same affordance converge together.

each $(effect, (object, action))$ tuple, our model learns the equivalence between the objects. The two equivalences learned by our model can be shown by our notation as the following,

- $(inserted, (< insertable\text{-}openings >, insert)$
- $(not\text{-}inserted, (< non\text{-}insertable\text{-}openings >, insert)$

Where $< insertable\text{-}openings >$ represents all openings that are insertable by the rod attached to the robot and $< non\text{-}insertable\text{-}openings >$ represents all openings that are not insertable by the rod attached to the robot.

B. Graspability

While the previous experiment showed that our model can encode single-agent affordances, this experiment aims to show our model can be used to encode multi-agent affordances and thus can be used for cross-embodiment learning. Furthermore, in the previous experiment, the affordance used depended primarily on the object. To address these, in this experiment, we showed that our experiment can learn multi-agent affordances that depend not only on the object but also on the action. The task is defined as grasping and lifting an object on the table.

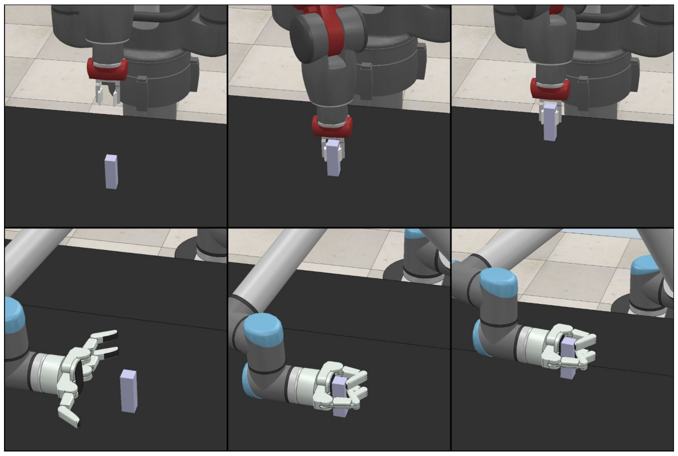

Fig. 7. The actions used in the experiment conducted in Section IV-B. The grasp action of Baxter can be seen at the top and the grasp action of UR-10 can be seen at the bottom of the figure.

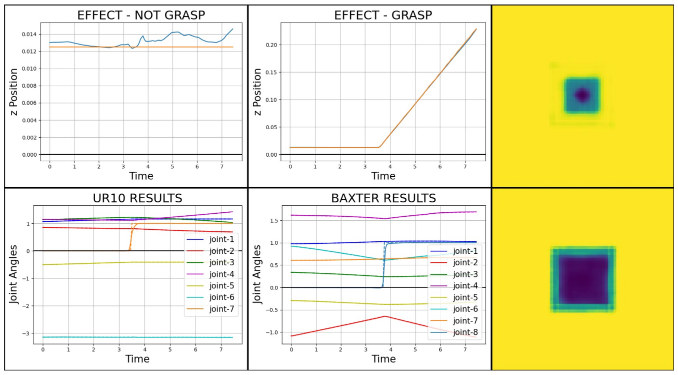

Fig. 8. The results of the grasp experiment conducted in Section IV-B. The difference between the generated and actual effect trajectories can be seen at the top left and center of the figure where orange lines represent the ground truth and blue lines represent the predicted trajectories. The difference between the generated and actual action trajectories can be seen at the bottom left and center of the figure where the dashed lines represent the ground truth values and the solid lines represent the model predictions. The depth image at the top right represents the output image when the given input image is the image of an object that can be grasped by Baxter, and the bottom is the output when the given input image is an image of an object that can not be grasped by Baxter.

Two robots (UR-10 with BarrettHand and Baxter with Baxter gripper) are used in a simulated experiment. The actions used in this experiment can be seen in Figure 7. Similar to the previous experiment, varying-sized objects are used. Among the 10 objects that are used, 5 can be graspable only by UR-10, and the remaining 5 can be grasped and lifted by both robots (since the opening of the Baxter gripper is small it can not grasp large objects). Out of these 10 objects, 8 are used for training and 2 are used for testing with similar constraints to the previous experiment. The effect is defined as the change in the z coordinate of the object. If the object is grasped and lifted, the z coordinate changes; if not, it does not change.

The results of this experiment can be seen in Figure 8. The difference in effect output when the given inputs represent a graspable condition or non-graspable condition can be seen in the top left and center of the figure. Note that the scale of the grasp effect result plot and the non-grasp effect result plot are different, which is why the error on the left of the figure looks higher. Similar to the previous experiment, the system learned common representations for objects that behave similarly. The outputs of the model when such objects are given to the system can be seen on the right side of the figure. The depth image on the bottom is the output when the inputs include objects that can not be grasped and lifted by Baxter, and the image at the top is the output when the inputs include objects that can be grasped and lifted by both robots. The results also do not change when the images in the test set are given. The generated action trajectories can be seen at the bottom left and center of the figure.

C. Rollability

Another affordance we investigated is object rollability. Two robots (KUKA LBR4+ and UR-10) with two different grippers (Robotiq85 and BarrettHand) are used as agents to conduct this experiment. The task is defined as pushing the given objects in 3 different directions (to the left, to the right, and straight). 2 different actions are used for UR-10 (one with its fingers while the gripper is closed and the other with the palm of the gripper). Push-to-left actions with the cuboid object can be seen in Figure 9. The other actions are realized similarly with objects in the same position.

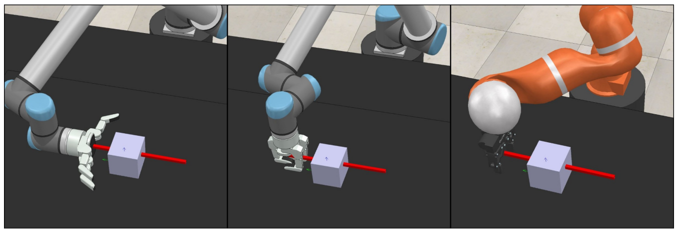

Fig. 9. The setup of the experiment conducted in Section IV-C. All images are from push-to-left actions. The left of the figure shows the push action of UR-10 with the palm of the open gripper, the one in the center shows the push action of UR-10 with the fingers of the closed gripper, and the one on the right shows the push action of the KUKA LBR4+ robot. Red lines represent the path the robots' end effectors follow to push the objects. Push to right and push straight actions are similar. The objects are always centered.

Objects with different rollability characteristics are used in the experiment. These objects can be seen in Figure 10. While the rollability of some of them does not depend on the action applied to them (cuboid, upright cylinder, cone, and sphere), the rollability of the others (cylinders standing on their side) depends on the direction of the action applied to them. All objects except the cone and the upright cylinder are used during training, and the cone and the upright cylinder are used to check whether our system can transfer affordances from one agent to another(s).

The results of this experiment can be seen in Figure 11. The difference between the effect trajectory of a rollable object and a non-rollable object can be seen at the top left and center of the figure. When the object is not rollable the object stops when the robot stops pushing (on the left), and when the object is rollable, the object keeps moving after the robot stops pushing (in the center). The generated effect trajectories in the figure can be obtained by giving the system a

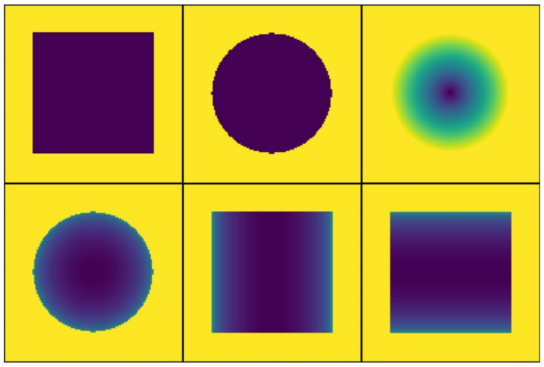

Fig. 10. The depth images of rollable and non-rollable objects used in the experiment in Section IV-C. The images at the top are non-rollable (from left to right: cuboid, upright cylinder, and cone). The image at the bottom left is always rollable (sphere), and the rollability of the objects with the images in the bottom center and right depends on the direction of the push action applied to them.

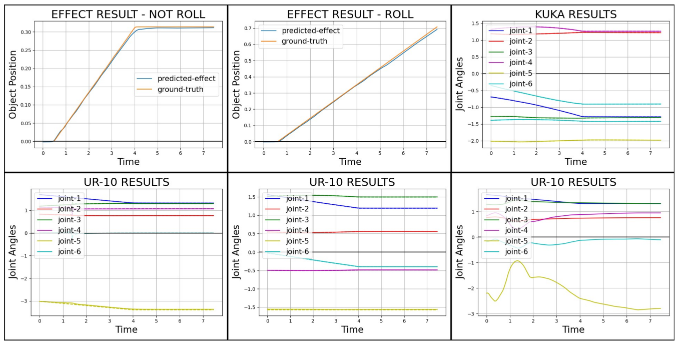

Fig. 11. The results of the rollability experiment conducted in Section IV-C. The difference between the generated and actual effect trajectories can be seen at the top left and center of the figure where orange lines represent the ground truth and blue lines represent the predicted trajectories. The generated and actual action trajectories of the KUKA robot, open gripper push action of UR-10, and closed gripper push action of UR-10 can be seen at the top right, bottom left, and bottom center of the figure, respectively. The generated action trajectory of UR-10 when no information about the UR-10 action is given to the system can be seen at the bottom right of the figure.

rollable or non-rollable input combination based on the desired trajectory. Unlike the experiment in Section IV-B, since the rollability of the object does not depend on the action, these effect trajectories can be generated even when only the object is given as input. Thus, our model can predict the object's rollability solely based on its depth image.

Aside from effect trajectories, our system can reproduce the action trajectories of the robots with almost negligible error, as seen from the top right, bottom left, and bottom center of the figure. However, since UR-10 has two different actions with the same outcome in this experiment when no information regarding its action is given, the system is unable to generate a valid action trajectory for UR-10. One of such trajectories can be seen in the bottom right of Figure 11. It can be seen that the generated trajectories are very different from the valid ones (the ones on the right and at the center). We believe this is another case where selective loss is useful. After training normally for several epochs, we continued training the model with selective loss used in the UR-10 output of the system. During testing, we saw that not only did the system continue to generate the correct action trajectories for UR-10 when information about its action was given to the system but also it was able to output one of the valid actions when no information about its action is given to the system.

In this experiment, we also checked the system's ability to transfer a learned affordance from one robot to the other. To achieve this, after training the model with cuboid, sphere, and cylinders standing on the side objects, we trained the model with the remaining objects (upright cylinder and cone) with only one robot's one of actions. For this, we devised a two-step training procedure. First, we train the system with the new data (cone and cylinder objects) for one training step. Then, to prevent catastrophic forgetting, we train the system with the old data for one training step. This procedure is repeated for several epochs. In the end, when a new object image and the action of the robot the new object is not trained for is given to the system, it is able to predict the correct effect trajectory for that object. Furthermore, the system was able to generalize to the actions of other directions. In the trials, we trained the model with the new data using a push-to-left action, and it was able to generalize to the push-to-right and push-straight actions of both robots. However, this behavior is not always desirable since if the new data contains an object that behaves differently based on the direction it was pushed (like the cylinder that stands on the side), it would generalize incorrectly.

D. Real Robot Experiments

Since our system can generate action trajectories using effect trajectories and object information, it can be used for direct imitation. To test this ability, we devised an experiment using a real UR-10 with a Robotiq 3-finger adaptive gripper. The task is defined as pushing the given objects to different angles. The locations of the objects are tracked using a tabletop Intel RealSense depth camera which is also used to take the depth images of the objects used in the experiment. During testing, one of the objects is pushed to different angles by the experimenter. The final position of the object is given to the system along with the object's depth image. After putting the object back in the initial position, the action executed by the real robot was able to push the object to the position the experimenter pushed it to.

V. Conclusion

In this paper, we proposed and verified a model that can learn affordance representations that can be used to generate continuous action and effect trajectories and object images that constitute those affordances. For the affordances we used in the experiments, our system was able to learn common representations for equivalent representations. We analyzed the formation of these representations using a latent space analysis. We also showed that affordances could be used as a way to couple the actions of different robots, allowing cross-embodiment transfer. We also demonstrated our model's generalization capabilities in transferring affordance information between agents. Moreover, in real-world experiments, we showed that our model can be used for direct imitation. Finally, we introduced selective loss both for our system and vanilla CNMP systems and showed cases where it would be beneficial to use it.

VI. Acknowledgments

References

[1] E. Şahin, M. Cakmak, M. R. Doğar, E. Uğur, and G. Üçoluk, "To afford or not to afford: A new formalization of affordances toward affordance-based robot control," *Adaptive Behavior*, vol. 15, no. 4, pp. 447–472, 2007.

[2] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[3] T. Asfour, P. Azad, F. Gyarfas, and R. Dillmann, "Imitation learning of dual-arm manipulation tasks in humanoid robots," *International journal of humanoid robotics*, vol. 5, no. 02, pp. 183–202, 2008.

[4] H. B. Amor, O. Kroemer, U. Hillenbrand, G. Neumann, and J. Peters, "Generalization of human grasping for multi-fingered robot hands," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 2043–2050.

[5] M. Mühlig, M. Gienger, and J. J. Steil, "Interactive imitation learning of object movement skills," *Autonomous Robots*, vol. 32, pp. 97–114, 2012.

[6] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Using probabilistic movement primitives in robotics," *Autonomous Robots*, vol. 42, pp. 529–551, 2018.

[7] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 763–768.

[8] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal, "Online movement adaptation based on previous sensor experiences," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 365–371.

[9] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent service robotics*, vol. 9, pp. 1–29, 2016.

[10] Y. Huang, L. Rozo, J. Silvério, and D. G. Caldwell, "Kernelized movement primitives," *The International Journal of Robotics Research*, vol. 38, no. 7, pp. 833–852, 2019.

[11] S. Schaal, "Dynamic movement primitives-a framework for motor control in humans and humanoid robotics," in *Adaptive motion of animals and machines*. Springer, 2006, pp. 261–280.

[12] Y. Zhou and T. Asfour, "Task-oriented generalization of dynamic movement primitive," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3202–3209.

[13] C. Atkeson, A. Moore, and S. Schaal, "Locally weighted learning for control. lazy learning," 1997.

[14] A. Kramberger, A. Gams, B. Nemec, D. Chrysostomou, O. Madsen, and A. Ude, "Generalization of orientation trajectories and force-torque profiles for robotic assembly," *Robotics and autonomous systems*, vol. 98, pp. 333–346, 2017.

[15] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 800–815, 2010.

[16] S. Vijayakumar and S. Schaal, "Locally weighted projection regression: An o (n) algorithm for incremental real time learning in high dimensional space," in *Proceedings of the seventeenth international conference on machine learning (ICML 2000)*, vol. 1. Morgan Kaufmann, 2000, pp. 288–293.

[17] S. Calinon, P. Evrard, E. Gribovskaya, A. Billard, and A. Kheddar, "Learning collaborative manipulation tasks by demonstration using a haptic interface," in *2009 International Conference on Advanced Robotics*. IEEE, 2009, pp. 1–6.

[18] A. Pervez and D. Lee, "Learning task-parameterized dynamic movement primitives using mixture of gmms," *Intelligent Service Robotics*, vol. 11, no. 1, pp. 61–78, 2018.

[19] V. Chu, I. McMahon, L. Riano, C. G. McDonald, Q. He, J. M. Perez-Tejada, M. Arrigo, N. Fitter, J. C. Nappo, T. Darrell *et al.*, "Using robotic exploratory procedures to learn the meaning of haptic adjectives," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 3048–3055.

[20] H. Girgin and E. Ugur, "Associative skill memory models," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 6043–6048.

[21] D. Lee and C. Ott, "Incremental kinesthetic teaching of motion primitives using the motion refinement tube," *Autonomous Robots*, vol. 31, pp. 115–131, 2011.

[22] E. Ugur and H. Girgin, "Compliant parametric dynamic movement primitives," *Robotica*, vol. 38, no. 3, pp. 457–474, 2020.

[23] A. Droniou, S. Ivaldi, and O. Sigaud, "Deep unsupervised network for multimodal perception, representation and classification," *Robotics and Autonomous Systems*, vol. 71, pp. 83–98, 2015.

[24] A. Gams, A. Ude, J. Morimoto *et al.*, "Deep encoder-decoder networks for mapping raw images to dynamic movement primitives," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 5863–5868.

[25] A. Pervez, Y. Mao, and D. Lee, "Learning deep movement primitives using convolutional neural networks," in *2017 IEEE-RAS 17th international conference on humanoid robotics (Humanoids)*. IEEE, 2017, pp. 191–197.

[26] F. Xie, A. Chowdhury, M. De Paolis Kaluza, L. Zhao, L. Wong, and R. Yu, "Deep imitation learning for bimanual robotic manipulation," *Advances in neural information processing systems*, vol. 33, pp. 2327–2337, 2020.

[27] M. Y. Seker, M. Imre, J. H. Piater, and E. Ugur, "Conditional neural movement primitives." in *Robotics: Science and Systems*, vol. 10, 2019.

[28] M. Garnelo, D. Rosenbaum, C. J. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. J. Rezende, and S. Eslami, "Conditional neural processes," *arXiv preprint arXiv:1807.01613*, 2018.

[29] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey." *Journal of Machine Learning Research*, vol. 10, no. 7, 2009.

[30] Z. Zhu, K. Lin, A. K. Jain, and J. Zhou, "Transfer learning in deep reinforcement learning: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[31] W. Yu, J. Tan, C. K. Liu, and G. Turk, "Preparing for the unknown: Learning a universal policy with online system identification," *arXiv preprint arXiv:1702.02453*, 2017.

[32] T. Chen, A. Murali, and A. Gupta, "Hardware conditioned policies for multi-robot transfer learning," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[33] J. Yang, D. Sadigh, and C. Finn, "Polybot: Training one policy across robots while embracing variability," *arXiv preprint arXiv:2307.03719*, 2023.

[34] P. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, and W. Zaremba, "Transfer from simulation to real world through learning deep inverse dynamics model," *arXiv preprint arXiv:1610.03518*, 2016.

[35] Q. Zhang, T. Xiao, A. A. Efros, L. Pinto, and X. Wang, "Learning cross-domain correspondence for control with dynamics cycle-consistency," *arXiv preprint arXiv:2012.09811*, 2020.

[36] G. Konidaris and A. Barto, "Autonomous shaping: Knowledge transfer in reinforcement learning," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 489–496.

[37] A. A. Rusu, M. Večerík, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-real robot learning from pixels with progressive nets," in *Conference on robot learning*. PMLR, 2017, pp. 262–270.

[38] X. Liu, D. Pathak, and D. Zhao, "Meta-evolve: Continuous robot evolution for one-to-many policy transfer," in *The Twelfth International Conference on Learning Representations*, 2023.

[39] M. Akbulut, E. Oztop, M. Y. Seker, X. Hh, A. Tekden, and E. Ugur, "Acnmp: Skill transfer and task extrapolation through learning from demonstration and reinforcement learning via representation sharing," in *Conference on Robot Learning*. PMLR, 2021, pp. 1896–1907.

[40] H. Aktas, Y. Nagai, M. Asada, E. Oztop, and E. Ugur, "Correspondence learning between morphologically different robots via task demonstrations," *arXiv preprint arXiv:2310.13458*, 2023.

[41] E. Rohmer, S. P. N. Singh, and M. Freese, "Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, 2013.