Learning Affordances from Interactive Exploration using an Object-level Map

Paula Wulkop[1*], Halil Umut Özdemir[1*] Antonia Hüfner[1], Jen Jen Chung[2], Roland Siegwart[1], and Lionel Ott[1]

[1] Autonomous Systems Lab, ETH Zurich, Switzerland
`pwulkop@ethz.ch, halilumutozdemir10@gmail.com`
[*] Equal contribution.
[2] School of EECS, The University of Queensland, Australia

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017008 (Harmony) and under the Marie Skłodowska-Curie grant agreement No 953454 (AERO-TRAIN).

Abstract. Many robotic tasks in real-world environments require physical interactions with an object such as *pick up* or *push*. For successful interactions, the robot needs to know the object's affordances, which are defined as the potential actions the robot can perform with the object. In order to learn a robot-specific affordance predictor, we propose an interactive exploration pipeline which allows the robot to collect interaction experiences while exploring an unknown environment. We integrate an object-level map in the exploration pipeline such that the robot can identify different object instances and track objects across diverse viewpoints. This results in denser and more accurate affordance annotations compared to state-of-the-art methods, which do not incorporate a map. We show that our affordance exploration approach makes exploration more efficient and results in more accurate affordance prediction models compared to baseline methods.

Keywords: Affordances, Object-level mapping, Interactive Exploration

1 Introduction

Mobile robots operating in human spaces often need to interact with objects, for example for a task such as finding and retrieving a laptop, the robot needs to pick up the laptop before placing it on top of a table. To perform such manipulations of the environment, a robot must know the potential interactions it can perform with the objects in the world, which are known as affordances [1]. Most recent works on visual affordance estimation are supervised learning approaches that rely on pre-existing single-object affordance datasets [2–6]. In contrast, we take the view that affordances for interactions are not only an object property but also depend on the robot's intrinsic capabilities [7]. For example, whether an object is pickupable depends on the strength and the gripper type of the robot arm. Therefore, a model learned using the experiences of one robot is not directly

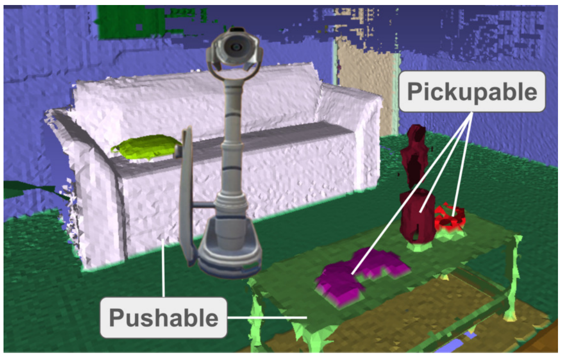


Fig. 1: An object-level map of a living room scene generated by TSDF++ [8] with the robotic agent from the iTHOR simulation framework [9].

applicable to another robot with a different morphology. Additionally, to obtain good performance transfer between training and testing, the training data should ideally consist of objects contextualized in the environment and not of isolated objects, as is the case in most existing affordance datasets. Consequently, this work explores how to leverage self-supervised learning to create a robot-specific, scene-level affordance dataset without relying on manual labels.

With this in mind, a promising approach is to learn affordances from data generated by a robot interacting with its environment in a photo-realistic simulator. Recently, Nagarajan and Grauman [10] proposed such an approach for learning affordances from interactions. They use deep reinforcement learning (RL) to explore a new 3D environment while, in parallel, training an affordance model based on the results of the executed interactions. The authors showed that affordance-based exploration leads to faster discovery of interactable regions. Additionally, their learned affordance model proved helpful in complex downstream tasks like washing the dishes. However, since the exploration framework of Nagarajan and Grauman [10] operates on each frame individually, it is unable to exploit structural scene information to expedite the agent's affordance exploration and learning, i.e. the learned exploration agent has no sense of object permanence. In contrast, modern robotic mapping pipelines provide representations at the object level [8, 11] and explicitly offer dense object segmentations that can be associated across multiple viewpoints. This information can be directly applied to extrapolate observed interaction data across the full exploration sequence and substantially improve learning efficiency by providing higher interaction success rates.

In this work, we create and use an explicit map of the environment in which each discovered object and interaction is stored (see Fig. 1). We integrate TSDF++ [8], a dynamic object-level mapping framework, into the pipeline to enable the agent to identify different object instances and track their movements across different viewpoints. Our approach consists of an RL agent exploring the simulated training environments by interacting with various objects and storing this data in the object-level map. From this, we periodically extract a dataset to concurrently train an affordance classifier that learns to accurately predict agent-specific object affordances. While the RL policy ensures that the robot efficiently explores the scene during training, the final output of the system is the learned affordance predictor.

Our contribution is the introduction of an object-level map into an interactive affordance learning pipeline. The benefits of an explicit map representation are twofold: Firstly, storing the interaction experiences in the object-level map permits the propagation of interaction outcomes across different viewpoints, creating more accurate and complete annotations. Secondly, explicitly including object information as state information improves the interaction success rate and efficiency, leading to higher-quality training data. Overall, we show that the usage of an object-level map improves the quality of the affordance predictions, therefore allowing a robot to explore a new scene more efficiently.

2 Related work

2.1 Visual Affordance Learning

Affordances are defined as the actions that an agent can apply to an object [1, 12]. They are usually learned from visual inputs by either segmenting the objects and extracting the parts that are relevant for an interaction [13, 14], or by performing dense, pixel-wise affordance predictions [2–5]. Most state-of-the-art approaches focus on affordance learning via supervised learning, for which annotated datasets of object affordances are required [2–5]. However, most datasets only contain single-object affordances [2, 6, 15, 16] and do not consider the context in a scene. Additionally, due to the physical constraints of each robot, affordance definitions can vary from robot to robot. Therefore labels extracted either from human annotation or from demonstrations [17–22] do not always generalize across different robots. Rather than learning affordances from annotated datasets, we therefore propose to learn affordances from data generated by an agent interacting with different objects in a scene-level simulation environment.

2.2 Interactive Exploration

Most works on the exploration of unknown environments focus on navigation actions, using either learning-based methods like reinforcement learning [23–27] or classical SLAM-based methods [28, 29]. Recently, Liang *et al.* [30] proposed an embodied learning framework that uses the data collected during active exploration of a 3D environment to learn visual representations. However, not only navigation but also interactions with objects, e.g. opening the fridge or picking up a pillow, are important for using robots in our daily lives. Recently developed simulators with high-level interaction capabilities [9, 31, 32] enable pipelines where agents can follow instructions or answer questions that require interaction [32–35]. While these works focus on task-driven goals, we aim to use the interactions in order to build up an explicit knowledge of affordances.

2.3 Map Representations for Exploration

A structured representation of visited locations is beneficial for effectively exploring a new environment. Gervet *et al.* [36] show that the usage of explicit maps as a state compared to an implicit memory improves the generalization performance of the agent. [25, 26] use a top-down occupancy map to keep track of explored areas and the obstacles in them. In [37] 3D information is crucial since the goal is to find a specific object in a scene, so they use fused 3D point clouds as observations. Similarly to our approach, Chaplot *et al.* [38] build a 3D semantic map for self-supervised label propagation. Specifically, the labels are stored in a 3D semantic map and then projected onto the agent's frame-wise observations from different viewpoints, thereby generating pixel-wise instance labels for new frames. In this work, we will use a similar approach to generate additional affordance labels.

2.4 Learning Affordances from Interactions

In previous works, interactive exploration has been used to learn affordances for simple objects in table-top environments [39, 40] and for block pushing tasks [41]. Wang *et al.* [42] focus on learning affordances for moving obstacles out of the way when navigating through a cluttered scene. Most relevant to our work, Nagarajan and Grauman [10] recently proposed a method to interactively learn generic object affordances on a scene-level. Their approach consists of a reinforcement learning algorithm that learns the interactive exploration strategy and an affordance network that is trained with the interactions executed during exploration. For each executed interaction, they mark the point of interaction and back-project it to the previous frames in order to annotate objects from different viewpoints. However, since they do not create an explicit map, they cannot re-identify an object instance, leading to very sparse annotations and a low interaction success rate.

3 Approach

Our approach aims to train an affordance model using an agent's previous experiences, which are stored in an object-level map. Fig. 2 shows an overview of our approach. The first component on the top shows the simulation environment where the RL agent can collect experiences by moving around and attempting to either *pick up* or *push* the objects in the scene. The simulator then returns the result of whether the interaction was successful or not. Once an RL episode finishes, the collected experience, in conjunction with the object level map and the RGB-D data from the onboard sensor, is used to generate new labels to improve the affordance network. In the following, we provide detailed descriptions of these components.

3.1 Mapping Framework

An object-level map is used to reconstruct the environment during exploration. At every step, the RGB-D frame and the instance segmentation mask are used

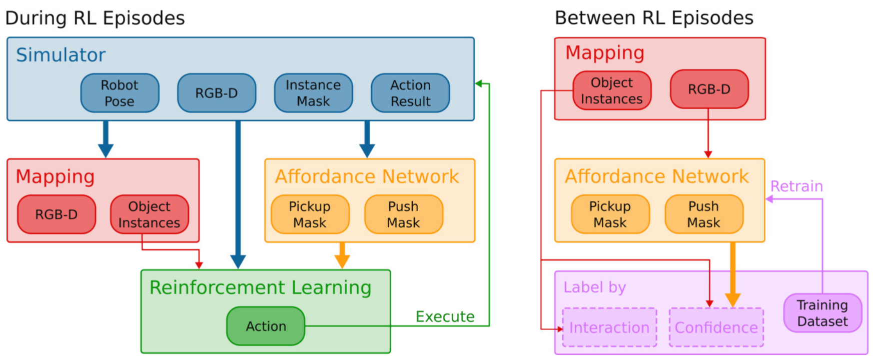


Fig. 2: **Overview of our method during training.** At each time step, an action is executed and the simulator (blue) outputs if the action was successful, as well as the RGB-D image, ground truth instance segmentation mask, and robot pose. The mapping module (red) updates the map with this data, while the affordance module (yellow) predicts the affordances. The RL exploration policy module (green) estimates the next optimal action using the current state of the object-level map, RGB-D image, and affordance estimation as input. At the end of the episode, the label module (purple) annotates each object instance based on the interaction data and estimations from the current affordance network. Finally, the affordance network is retrained after every episode with the new data.

to update the map. Additionally, the affordance label is stored in the map for those objects which have already been annotated. We use TSDF++ [8] which is a voxel-based mapping framework that models each object instance as a separate truncated signed distance function (TSDF) layer. An important feature of TSDF++ is that it is a dynamic object-level framework, allowing it to track object movements when the agent interacts with an object. The advantage of tracking the object is that the affordance label can be applied to the same object observed from different viewpoints. One limitation of TSDF++ is that it can only be used with rigid objects, i.e. affordances such as *slice* or *open* cannot be represented in the map. With a more flexible mapping framework however, the approach presented here would also extend to such actions.

3.2 Interactive Exploration Policy

We model this exploration task as a Markov decision process and use reinforcement learning to learn the best policy. The action space, state space, and reward are described in the following.

State space The state space of the agent is a combination of the raw perception input, affordance estimation, spatial representations, and action history. Fig. 3 shows the visual representation of the state space. From the onboard sensor the RGB image as well as the depth image are used, from which we create a so-called distance image by projecting the depth image into the robot arm's base frame to indicate if an object is within reach. The output of the current affordance estimation network indicates whether an object is likely to be interactable. Additionally, the state space contains the action history of the last five actions with

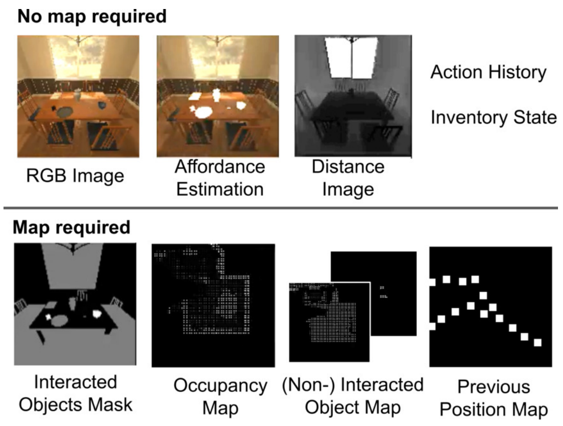


Fig. 3: **Visual representation of the state space.** The elements in the top row do not require a map, while the states in the bottom are obtained through the map.

their obtained success and reward as well as the inventory indicating if the robot currently holds a picked up object. In addition to these states derived from the current RGB-D input, we add states based on the object-level map to improve exploration efficiency. An egocentric 2D grid map of the previously visited positions as well as the interacted objects mask with the labels *interaction successful*, *interaction unsuccessful* and *no interaction attempted* are used as states. Furthermore, we create an occupancy map and an interacted and a non-interacted objects map by projecting the 3D voxels into 2D. While the compression into 2D space, which is necessary to ensure computational feasibility of the RL pipeline, inevitably leads to a loss of information about the vertical arrangement of objects, it captures the planar layout which is crucial for navigation. All maps are used once as global maps with a size of 10 m $\times$ 10 m and once as local egocentric maps of size 2 m $\times$ 2 m.

Action space The action space of the robot in the iTHOR [9] simulation environment consists of navigation actions and interactions with objects. The discrete navigation actions are: *forward movement by* 0.25 m, *rotate right by* 30°, *rotate left by* 30°, *look up by* 15°, *look down by* 15°. The possible interactions with objects are *pick up*, *drop* and *push*. Once the robot picks up an object, it can attempt to move the object by ± 0.1 m along the three axes.

Interaction selection For every robot pose there are usually multiple objects that the robot could try to interact with. [10] solve this ambiguity by always selecting the object that is most centered in the current RGB frame. We however want to enable the robot to actively select the object to interact with as we believe that this will lead to a more targeted exploration. Since the number of objects varies from frame to frame but the RL action space has a fixed size, we implemented the following approximation for the object selection: The center region of the image frame is split up in a grid of nine equal-sized cells and the objects are assigned to the grid cell that they are closest to. In this way, the robot selects the object by selecting the corresponding grid cell, where empty grid cells cannot be selected due to a mask and for cells with multiple objects the largest object will be selected. Additionally, we use the output of the affordance network to filter out objects with a low confidence score.

Reward To train the agent, we use a reward function with three components, as shown in (1). R_{nav} encourages spatial exploration by rewarding each newly visited position as well as each new orientation at a previously visited position. R_{int} rewards each interaction with a new object instance, preventing exploitation of known objects. R_{fail} punishes each failed action, for example if the agent tries to pick up a couch which is non-pickupable, discouraging repeat negative interactions.

$$R = \alpha_1 R_{\text{nav}} + \alpha_2 R_{\text{int}} + \alpha_3 R_{\text{fail}} \tag{1}$$

$$R_{\text{nav}} = \begin{cases} 1 & \text{if novel position} \\ 0.3 & \text{if novel orientation at prev. pose} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$R_{\text{int}} = \begin{cases} 1 & \text{if successful new interaction} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$R_{\text{fail}} = \begin{cases} -1 & \text{if action failed} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Exploration policy learning The policy network is trained using Proximal Policy Optimization (PPO) [43] with rollouts of 600 time steps. All the elements of the state space are encoded using 2D convolutional encoders for the image-like inputs with a dimension of (128, 128) and MLPs for 1D states. These encodings are concatenated and then passed through two three-layer MLPs to separately predict the action function and the value function, with MLP layer dimensions of (64, 32, 21) and (64, 32, 1), respectively. Additionally, to obtain faster convergence, we apply an action mask which disables actions that are infeasible given the current state, e.g. the action *drop* is disabled if the agent does not hold any object.

3.3 Affordance Learning

The affordance estimation network is trained on the interaction data collected during exploration, as shown in Fig. 2. In the following, we will explain how the interaction data collected by the robot during an episode is labeled and used to retrain the affordance model. The improved affordance model is then used in the exploration pipeline, thereby ensuring that the two components are helping each other to converge.

Data generation and labeling To generate data, the exploration policy executes at each RL time step an action and the simulator returns whether it was successful or not. Rather than annotating the object directly on a single RGB-D frame after each interaction, we annotate the object instance in the map. This method is similar to SEAL [38] which also uses a map to propagate annotations. The labelled map allows us to annotate each frame of the episode in which this object instance has been visible, thereby applying the information of a single interaction to multiple frames. For each frame we use the segmentation mask to create dense labels for all pixels of an annotated object. In our implementation, the segmentation mask comes directly from the simulator, but it could also easily be obtained by a separate state-of-the-art segmentation network, e.g. [44]. The baseline [10] instead does not use a segmentation mask and instead labels a sphere of radius 20 cm centered on the interaction position (see Fig. 4). Since this sphere is of a fixed size, this strategy can lead to false positive labels for small objects and false negative labels for large objects.

Overall, the instance mapping allows us to consistently annotate objects across different viewpoints and robot movements and thereby creating more densely labeled datasets. The first and most straightforward annotation approach is to perform *annotation by interaction*, meaning that if the agent performed a successful or unsuccessful interaction with an object instance, it is annotated accordingly. However, since these interactions are sparse, we propose to additionally apply a self-supervised annotation strategy which we call *annotation by confidence*. This approach takes advantage of the classifier likelihoods outputted by the affordance network to label additional data. After the affordance network starts to converge, the affordance prediction is queried for each frame F_i and is used to further annotate objects with which the robot has not interacted. For each frame F_i in which the object appears, the percentiles of the affordance predictions are computed. If the maximum of the 95th percentile over all these frames is higher than a threshold $\xi_T = 0.9$, or if the minimum of the 5th percentile is lower than $\xi_F = 0.1$, then the object instance is annotated accordingly as a successful or unsuccessful interaction, i.e.:

$$\text{label} = \begin{cases} 1 & \text{if} \max\{\{F_i\}_{P95}\,\forall i\} > \xi_T \\ 0 & \text{if} \min\{\{F_i\}_{P5}\,\forall i\} < \xi_F \\ \text{none} & \text{otherwise.} \end{cases} \tag{5}$$

Note that since neural networks are known to be over-confident in their predictions, it could be beneficial to apply a calibration method, as suggested in [45]. At the end of each episode a partially annotated RGB-D affordance dataset is created by combining the object interaction annotations, RGB-D images and segmentation masks for each frame. In order to ensure a balanced dataset, we only use frames that contain pixels with both positive and negative labels.

Training procedure The affordance network is based on a U-Net architecture [46] with a loss function that is a combination of binary cross entropy and dice loss. The U-Net implementation contains an encoder and a decoder with 4 downscale layers each, with output sizes of (24, 48, 92, 164). To train the affordance network, we use the Adam optimizer with weight decay. After each episode, the newly generated dataset is split up into training, validation, and testing sets and added to a global affordance dataset. The training of the affordance network is then continued with this global dataset, whereby the dataset of an episode is removed from the global dataset after its 35th usage in the training loop. If the newly trained affordance network outperforms the previous one, it replaces it.

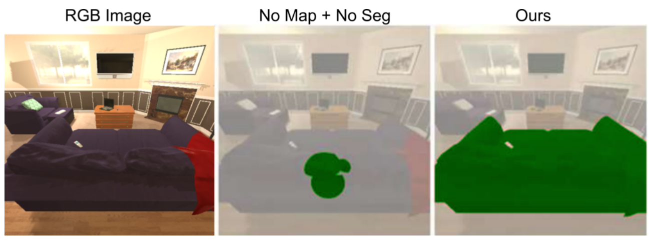


Fig. 4: An example image of the annotation approach used by [10] and the No Map + No Seg ablation (middle) compared to our approach which uses object segmentation and is therefore able to annotate the full object (right). Annotation masks are shown in green.

4 Experiments

In this section we evaluate the effect of using an object-level map for the affordance learning. First, we evaluate if it increases the efficiency of the policy learning. Second, we evaluate the accuracy of the resulting affordance model.

4.1 Experimental Setup

Simulation environment We evaluate our pipeline on the iTHOR [9] simulation environment, which supports high-level interactions with various types of objects in realistic 3D indoor scenes. We use living room scenes such as the one shown in Fig. 1 since they contain a large variety of objects that the agent can interact with and also have large areas to explore. The scenes contain up to 50 different object categories which support 0 to 2 interactions (*push*, *pick up*). We split the 30 living room scenes into 25 training scenes (20% of this data is used for validation) and five testing scenes.

Baseline We compare our method to the best-performing method in the state-of-the-art paper by Nagarajan and Grauman [10] which they call "IntExp(PT)". Note that they also evaluated a different version of their method that uses object segmentation instead of spherical labels ("IntExp(Obj)"), but this method was outperformed by "IntExp(PT)". For this reason, we compare against the "IntExp(PT)" approach, which we implemented in our framework and call it here IntExp [10]. The following adaptions were necessary to make their approach directly comparable to our results: We retrained their code on the two affordances *pick up* and *push*, since they had used additional affordances which are not covered by our approach. Additionally, we used the AI2-THOR living room environments instead of the kitchen and we increased the image size slightly from 80x80 pixels to 128x128 pixels, since these settings were also used in our approach. Finally, our computational hardware - which we also used to train our approach - limited us to run 8 parallel processes during the RL training instead of 16 as in the original implementation, while still keeping the total number of training steps the same.

Ablations To evaluate how the integration of an object-level map impacts the affordance prediction and the exploration rate, we compare our approach to two ablations of our full method:

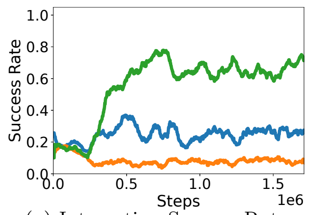
– Ours: The full method using the object-level map as described in Section 3. It uses the ground truth segmentation mask for labeling and applies annotation by interaction and confidence.
– No Map + Seg: This ablation evaluates the contribution of the map, therefore the state space contains only elements that do not require a map (RGB image, distance image, affordance estimates). As in IntExp [10], instead of a map we use a Gated Recurrent Unit (GRU) module in the RL part to aggregate observations over time.
– No Map + No Seg: This ablation aims to evaluate the importance of object segmentation. Instead of the ground truth segmentation mask used in our full approach, we apply the annotation strategy from IntExp [10]. Annotations from multiple viewpoints are generated by projecting 3D interaction points obtained during the episode (successful and unsuccessful) into all captured images where the object was visible shortly before and after the interaction. As objects have a volume, a sphere of radius 20 cm centered on these interactions is used to generate interaction labels.

4.2 Affordance-based Interactive Exploration

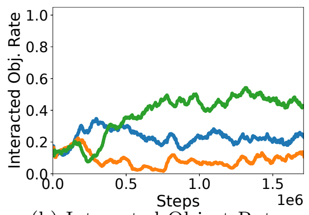
In this section, we address the question of whether the interactive exploration strategy is made more efficient by using an explicit map. To evaluate the performance during training of our pipeline compared to the ablation baselines without a map, we use the following metrics:

– Affordance IoU: Intersection over Union of affordance predictions compared to ground truth labels per frame.
– Object-level Accuracy: The ratio of objects with correctly predicted affordance labels over all visible objects in a frame. Object-level labels are obtain by performing a majority vote of all pixels within the object boundaries.
– Interaction Success Rate: The ratio of successful interactions over all attempted interactions during an episode.
– Interacted Object Rate: The ratio of the objects that the robot interacted with during an episode over all interactable objects in the scene.
– Interactable Annotation Rate: The ratio of pixels that are correctly annotated as *interactable* over all pixels.
– Non-interactable Annotation Rate: The ratio of pixels that are correctly annotated as *non-interactable* over all pixels.

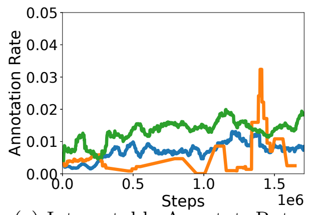
In each episode, the object positions, object states, initial position of the agent, and initial camera viewpoint are sampled randomly. At every time step, the agent executes an action from its action space (Section 3.2) and obtains feedback from the simulator on whether or not the action was successful. Fig. 5 shows the behavior of the system over the course of the RL training episodes. For our approach the agent not only interacts with more distinct objects per episode, but it also is more successful when trying out interactions with objects (Fig. 5a and b). This shows that our approach leads to a more targeted exploration which results in more informative labels. Interestingly, annotation by confidence, which is activated after an empirically determined number of 400 000



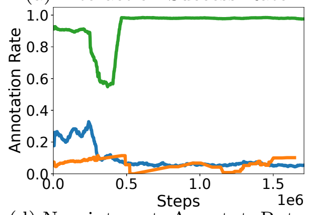
(a) Interaction Success Rate



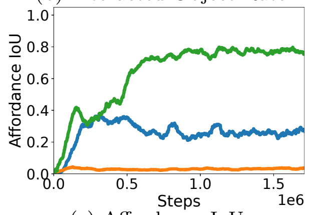
(b) Interacted Object Rate



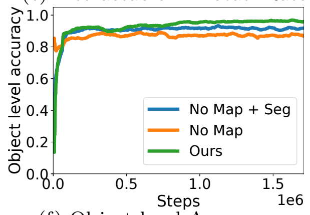
(c) Interactable Annotat. Rate



(d) Non-interact. Annotat. Rate



(e) Affordance IoU



(f) Object-level Accuracy

Fig. 5: The training curves for the *pick up* affordance show that our approach leads to a higher interaction success rate and a better affordance estimation performance.

steps, seems to boost the performance of the Interaction Success Rate since more informed interactions can be executed by leveraging the knowledge of the pre-trained affordance model.
Fig. 5c and d show that our approach leads to a higher ratio of annotated pixels compared to the approaches that do not accumulate information in a map. Our approach can annotate more pixels as *pickupable* for multiple reasons: Not only does our agent interact with more objects successfully, it also processes these annotations more efficiently by propagating them to multiple frames using the map. Similarly, the high non-interactable annotation rate of our approach is mostly caused by the annotation of large objects like the floor and walls which can be annotated either through interaction or through annotation by confidence and then re-identified in every frame thanks to the map. Overall this suggests that retaining knowledge about object instances, which is only possible when using a map, provides an advantage over just annotating images by backprojecting interaction points as done by [10].
Finally, we evaluate the quality of the resulting affordance predictions over time by looking at the pixel-level score Affordance IoU and the Object-level Accuracy (Fig. 5e and f). Our approach shows a faster and larger increase of the Affordance IoU than for the baselines, which means that the model quickly learns to predict the affordances accurately. The Object-level Accuracy converges quickly to a high level in all approaches, which is partially due to the fact that the predictions on an object level are more forgiving to pixel misclassifications.
To visualize the annotation quality, Figure 6 shows a qualitative comparison between our approach and the ablation baselines. As highlighted by the cyan colored circle, the labeling of the No Map + No Seg baseline especially leads to wrong annotations for very small objects since the labeling sphere has a fixed size. The magenta colored circle highlights an object that is annotated in our approach but not in the baselines. This is due to the annotation by confidence process which leverages the pretrained affordance model to label new objects.

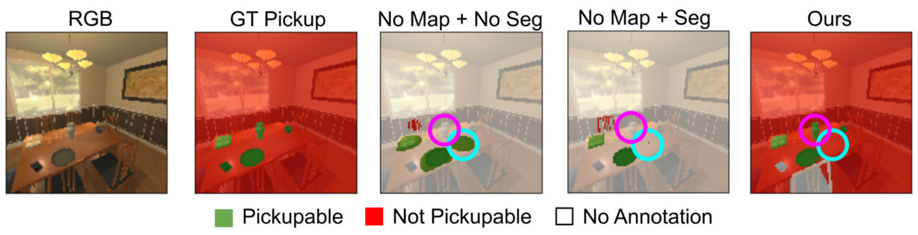


Fig. 6: *Pickupable* annotations for an example frame. The cyan circle shows a small pickupable object for which the No Map + No Seg baseline annotates a fixed big area since it does not use the segmentation mask. The magenta circle shows an object that the baseline methods fail to label but our approach is able to annotate by confidence.

Table 1: Affordance prediction (Median and [10th, 90th] percentile), averaged over 611 frames of the test scenes.

	IntExp [10]	No Map + No Seg	No Map + Seg	Ours
		Pick up		
Precision	0.03 [0.00, 0.22]	0.00 [0.00, 0.06]	0.10 [0.00, 0.75]	**0.20 [0.00, 0.99]**
Recall	0.07 [0.00, 0.51]	0.0 [0.00, 0.15]	0.22 [0.00, 0.96]	**0.37 [0.00, 0.99]**
F1	0.04 [0.00, 0.26]	0.00 [0.00, 0.08]	0.13 [0.0, 0.65]	**0.22 [0.00, 0.85]**
OAcc	0.86 [0.74, 1.00]	0.82 [0.69, 1.00]	0.87 [0.77, 1.00]	**0.89 [0.78, 1.00]**
		Push		
Precision	0.36 [0.06, 0.68]	0.30 [0.09, 0.52]	0.52 [0.15, 0.85]	**0.99 [0.23, 1.00]**
Recall	0.52 [0.24, 0.69]	**0.93 [0.52, 1.0]**	0.83 [0.12, 0.99]	0.48 [0.01, 0.96]
F1	0.41 [0.09, 0.63]	0.44 [0.14, 0.67]	0.58 [0.12, 0.83]	**0.60 [0.01, 0.96]**
OAcc	0.65 [0.47, 0.85]	0.70 [0.50, 0.86]	**0.80 [0.60, 0.92]**	0.76 [0.57, 0.92]

4.3 Affordance Prediction

In this section, we evaluate the performance of the learned affordance model in comparison to our ablations as well as to the state-of-the-art work IntExp [10]. To evaluate the frame-wise affordance predictions, we created a test dataset by manually steering the agent through five unseen iTHOR living room scenes and recording each frame of the trajectories, resulting in 611 frames. In addition to the previously introduced Object-level Accuracy (OAcc), we use Precision, Recall, and F1 Score. Table 1 shows that for the *pick up* affordance our approach outperforms the baseline and the ablations for all metrics. As expected, the IntExp [10] method and the No Map + No Seg ablation behave similarly, since the ablation was designed to mimic the assumptions of [10]. They obtain low scores in precision and recall, which can be explained by the fact that the pickupable objects are usually sparse and small and therefore difficult to learn with the sphere-based annotation strategy. The comparison of the ablations shows

Table 2: Interaction Success Rate, calculated over the 236 objects of the test scenes.

	IntExp [10]		No Map + No Seg		No Map + Seg		Ours	
	Pick up	Push	Pick up	Push	Pick up	Push	Pick up	Push
Accuracy	0.72	0.58	0.69	**0.76**	0.75	0.72	**0.81**	0.61
Precision	0.75	0.76	0.50	0.77	0.76	**0.94**	**0.85**	0.90
Recall	0.12	0.51	0.03	**0.91**	0.30	0.60	**0.47**	0.43
F1	0.21	0.61	0.05	**0.83**	0.43	0.73	**0.60**	0.58

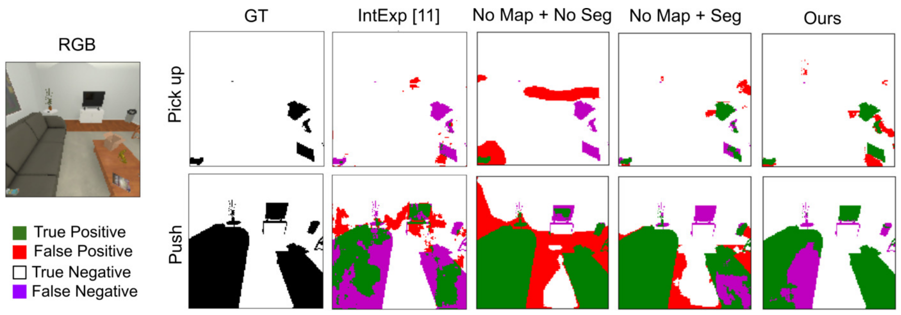


Fig. 7: Affordance predictions for an example test frame. For the *pick up* affordance our approach results in more true positives and fewer false negatives. Similarly, for the *push* affordance our approach results in fewer false positives.

clearly that the usage of both the segmentation masks and the map have a positive impact on all metrics. Fig. 7 shows the qualitative prediction results for one example test frame. The pickupable objects on the table are mostly correctly identified by our approach, whereas the IntExp [10] and the No Map + No Seg approaches misclassify all of these small pickupable objects. While our approach predicts some false positives on the table surface, those will tend to be averaged out when evaluating the predictions on an object-level. For the *push* affordance, Table 1 shows a trade-off between precision and recall. Our approach reaches a precision of 0.99, which means that it has a very low number of false positives. The baseline and the ablations on the other hand have a lower precision but a higher recall instead, meaning they are overly likely to predict *pushable*. This is clearly visible in Fig. 7, where the baseline approaches wrongly predict parts of the floor as pushable, while our approach only labels pixels that belong to pushable objects. Interestingly, in terms of object-level accuracy the No Map + Seg ablations outperforms our approach, which could be caused by the fact that for larger objects the misclassification of a few pixels is averaged out on an object-level.
Finally, we also evaluate the Interaction Success Rate to show whether the trained affordance model can correctly predict the feasibility of an interaction. We collected a dataset by placing the robot in front of every object in the test scenes and attempting the *pick up* and *push* actions, storing the outcome as ground truth labels. The affordance prediction modules are queried on the image frame and an object level prediction is calculated as the mean of the pixel-wise predictions of the object. In Table 2 we report the Accuracy, Precision, Recall and F1 score of the predicted object-wise interaction labels compared to the ground truth labels. The results show that for the *pick up* affordance our approach outperforms the baselines. For the *push* affordance, the No Map + No Seg baseline achieves a higher Recall and Accuracy than our approach, while the Precision is lower. Similar to the results from Table 1, this indicates that they are overly likely to predict *pushable*, which could be caused by the spherical annotations spilling over onto other objects.

5 Conclusion and Future work

In this work, we proposed a reinforcement learning pipeline to learn robot-centric affordances from interactions. Our approach allows an interactive exploration agent to learn how to efficiently explore an unseen environment while collecting a robot-specific dataset to train an object affordance model. To enable the agent to identify object instances and track object movements, we integrated an object-level map into our pipeline. This allows the agent to re-identify different object instances, thereby obtaining denser annotations from different viewpoints. Additionally, leveraging the information from the object-level map during exploration increases the interaction success rate and efficiency, leading to higher-quality training data. As a result, our method successfully learns to predict robot-centric object affordances.

The next step would be to transfer the learned affordance model to a real-world scenario. Given the domain gap of both the sensor data and the robot morphology, we envision a fine-tuning step in the real-world. We propose to use the pretrained affordance model and RL policy from simulation to collect additional interaction data in the real-world, using a modular robot-specific skill library to execute the high-level interaction commands. Another future step could be to introduce additional actions such as *break* or *open*. While our architecture in general is modular and easily allows adding additional affordances, our chosen TSDF++ representation would have to be developed further to also handle articulated, separable or deformable objects.

References

[1] J. J. Gibson, *The theory of affordances.* Hilldale, USA, 1977.

[2] A. Myers, C. L. Teo, C. Fermüller, and Y. Aloimonos, "Affordance detection of tool parts from geometric features," in *IEEE Int. Conf. on Robotics & Automation*, 2015.

[3] T.-T. Do, A. Nguyen, and I. Reid, "Affordancenet: An end-to-end deep learning approach for object affordance detection," in *IEEE Int. Conf. on Robotics & Automation*, 2018.

[4] S. Deng, X. Xu, C. Wu, K. Chen, and K. Jia, "3d affordancenet: A benchmark for visual object affordance understanding," in *Advances in Neural Information Processing Systems*, 2021.

[5] A. Nguyen, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis, "Detecting object affordances with convolutional neural networks," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2016.

[6] Z. Osama Khalifa and S. A. A. Shah, "Towards visual affordance learning: A benchmark for affordance segmentation and recognition," *arXiv e-prints*, 2022.

[7] J. J. Chung, J. Förster, P. Wulkop, N. Lawrance, L. Ott, and R. Siegwart, "It's just semantics: How to get robots to understand the world the way we do," in *Int. Symposium on Robotics Research*, 2022.

[8] M. Grinvald, F. Tombari, R. Siegwart, and J. Nieto, "TSDF++: A multi-object formulation for dynamic object tracking and reconstruction," in *IEEE Int. Conf. on Robotics & Automation*, 2021.

[9] E. Kolve *et al.*, “AI2-THOR: An Interactive 3D Environment for Visual AI,” *arXiv*, 2017.

[10] T. Nagarajan and K. Grauman, “Learning affordance landscapes for interaction exploration in 3d environments,” *Advances in Neural Information Processing Systems*, 2020.

[11] L. Schmid *et al.*, “Panoptic multi-tsdfs: A flexible representation for online multi-resolution volumetric mapping and long-term dynamic scene consistency,” in *IEEE Int. Conf. on Robotics & Automation*, 2022.

[12] L. Jamone *et al.*, “Affordances in psychology, neuroscience, and robotics: A survey,” *IEEE Transactions on Cognitive and Developmental Systems*, 2016.

[13] M. Kokic, J. A. Stork, J. A. Haustein, and D. Kragic, “Affordance detection for task-specific grasping using deep learning,” in *IEEE-RAS Int. Conf. on Humanoid Robotics (Humanoids)*, 2017.

[14] G. Li, D. Sun, L. Sevilla-Lara, and V. Jampani, “One-shot open affordance learning with foundation models,” in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2024.

[15] H. S. Koppula, R. Gupta, and A. Saxena, “Learning human activities and object affordances from rgb-d videos,” *The Int. Journal of Robotics Research*, 2013.

[16] A. Nguyen, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis, “Object-based affordances detection with convolutional neural networks and dense conditional random fields,” in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2017.

[17] Y. Zhu, C. Jiang, Y. Zhao, D. Terzopoulos, and S.-C. Zhu, “Inferring forces and learning human utilities from videos,” in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2016.

[18] T. Nagarajan, C. Feichtenhofer, and K. Grauman, “Grounded human-object interaction hotspots from video,” in *IEEE Int. Conf. on Computer Vision*, 2019.

[19] K. Fang, T.-L. Wu, D. Yang, S. Savarese, and J. J. Lim, “Demo2vec: Reasoning object affordances from online videos,” in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2018.

[20] J.-B. Alayrac, I. Laptev, J. Sivic, and S. Lacoste-Julien, “Joint discovery of object states and manipulation actions,” in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2017.

[21] T. Nagarajan, Y. Li, C. Feichtenhofer, and K. Grauman, “Ego-topo: Environment affordances from egocentric video,” in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2020.

[22] A. Delitzas, A. Takmaz, F. Tombari, R. Sumner, M. Pollefeys, and F. Engelmann, “SceneFun3D: Fine-Grained Functionality and Affordance Understanding in 3D Scenes,” in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2024.

[23] T.-L. Nguyen, D.-V. Nguyen, and T.-H. Le, “Reinforcement learning based navigation with semantic knowledge of indoor environments,” in *Int. Conf. on Knowledge and Systems Engineering (KSE)*, 2019.

[24] H. Tan, L. Yu, and M. Bansal, “Learning to navigate unseen environments: Back translation with environmental dropout,” in *Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.

[25] T. Chen, S. Gupta, and A. Gupta, “Learning exploration policies for navigation,” in *Int. Conf. on Learning Representations*, 2019.

[26] S. K. Ramakrishnan, D. Jayaraman, and K. Grauman, “An exploration of embodied visual exploration,” *Int. Journal of Computer Vision*, 2021.

[27] W. Qi, R. T. Mullapudi, S. Gupta, and D. Ramanan, “Learning to move with affordance maps,” in *Int. Conf. on Learning Representations*, 2020.

[28] B. Talbot, F. Dayoub, P. Corke, and G. Wyeth, “Robot navigation in unseen spaces using an abstract map,” *IEEE Transactions on Cognitive and Developmental Systems*, 2020.

[29] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov, “Learning to explore using active neural slam,” in *Int. Conf. on Learning Representations*, 2020.

[30] X. Liang, A. Han, W. Yan, A. Raghunathan, and P. Abbeel, “Alp: Action-aware embodied learning for perception,” *arXiv preprint arXiv:2306.10190*, 2023.

[31] X. Puig *et al.*, “Virtualhome: Simulating household activities via programs,” in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2018.

[32] X. Gao, R. Gong, T. Shu, X. Xie, S. Wang, and S.-C. Zhu, “Vrkitchen: An interactive 3d virtual environment for task-oriented learning,” *arXiv*, 2019.

[33] M. Shridhar *et al.*, “Alfred: A benchmark for interpreting grounded instructions for everyday tasks,” in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2020.

[34] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra, “Embodied question answering,” in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2018.

[35] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi, “Iqa: Visual question answering in interactive environments,” in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2018.

[36] T. Gervet, S. Chintala, D. Batra, J. Malik, and D. S. Chaplot, “Navigating to objects in the real world,” *Science Robotics*, vol. 8, no. 79, 2023.

[37] J. Zhang *et al.*, “3d-aware object goal navigation via simultaneous exploration and identification,” in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2023.

[38] D. S. Chaplot, M. Dalal, S. Gupta, J. Malik, and R. R. Salakhutdinov, “Seal: Self-supervised embodied active learning using exploration and 3d consistency,” *Advances in neural information processing systems*, vol. 34, pp. 13 086–13 098, 2021.

[39] L. K. Le Goff, O. Yaakoubi, A. Coninx, and S. Doncieux, “Building an affordances map with interactive perception,” *Frontiers in Neurorobotics*, vol. 16, 2022.

[40] J. Borja-Diaz, O. Mees, G. Kalweit, L. Hermann, J. Boedecker, and W. Burgard, “Affordance learning from play for sample-efficient policy learning,” in *IEEE Int. Conf. on Robotics & Automation*, 2022.

[41] D. Kim and G. Sukhatme, “Interactive affordance map building for a robotic task,” in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2015.

[42] X. Wang, Y. Liu, X. Song, Y. Liu, S. Zhang, and S. Jiang, “An interactive navigation method with effect-oriented affordance,” in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2024.

[43] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” 2017.

[44] A. Kirillov *et al.*, “Segment anything,” in *IEEE Int. Conf. on Computer Vision*, 2023.

[45] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, “On calibration of modern neural networks,” in *Int. Conf. on Machine Learning*, 2017.

[46] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *Medical Image Computing and Computer-Assisted Intervention*, 2015.