Learning Hybrid Actor-Critic Maps for 6D Non-Prehensile Manipulation

Anonymous Author(s)
Affiliation
Address
email

Abstract: Manipulating objects without grasping them is an essential component of human dexterity, referred to as non-prehensile manipulation. Non-prehensile manipulation may enable more complex interactions with the objects, but also presents challenges in reasoning about gripper-object interactions. In this work, we introduce Hybrid Actor-Critic Maps for Manipulation (HACMan), a reinforcement learning approach for 6D non-prehensile manipulation of objects using point cloud observations. HACMan proposes a *temporally-abstracted* and *spatially-grounded* object-centric action representation that consists of selecting a contact location from the object point cloud and a set of motion parameters describing how the robot will move after making contact. We modify an existing off-policy RL algorithm to learn in this hybrid discrete-continuous action representation. We evaluate HACMan on a 6D object pose alignment task in both simulation and in the real world. On the hardest version of our task, with randomized initial poses, randomized 6D goals, and diverse object categories, our policy demonstrates strong generalization to unseen object categories without a performance drop, achieving an 89% success rate on unseen objects in simulation and 50% success rate with zero-shot transfer in the real world. Compared to alternative action representations, HACMan achieves a success rate more than three times higher than the best baseline. With zero-shot sim2real transfer, our policy can successfully manipulate unseen objects in the real world for challenging non-planar goals, using dynamic and contact-rich non-prehensile skills. Videos can be found on the project website: https://hacman-2023.github.io.

Keywords: Action Representation, Reinforcement Learning with 3D Vision, Non-prehensile Manipulation

1 Introduction

The ability to manipulate objects in ways beyond grasping is a critical aspect of human dexterity. Non-prehensile manipulation, such as pushing, flipping, toppling, and sliding objects, is essential for a wide variety of tasks where objects are difficult to grasp or where workspaces are cluttered or confined. However, non-prehensile manipulation remains challenging for robots; previous work has only shown results with limited object generalization [1, 2] or limited motion complexity, such as planar pushing or manipulating articulated objects with limited degrees of freedom [3, 4, 5]. We propose a method that generalizes across object geometries while showing versatile interactions for complex non-prehensile manipulation tasks.

We present **H**ybrid **A**ctor-**C**ritic Maps for **Man**ipulation (HACMan), a reinforcement learning (RL) approach for non-prehensile manipulation from point cloud observations. The first technical contribution of HACMan is to propose an **object-centric action representation** that is **temporally-abstracted** and **spatially-grounded**. The agent selects a contact location and a set of motion parameters determining the trajectory it should take after making contact. The contact location is selected from the observed object point cloud which provides spatial grounding. At the same time, the robot decisions become more temporally-abstracted because we focus on only learning the contact-rich portions of the action.

The second technical contribution of HACMan is to incorporate the proposed action representation in an actor-critic RL framework. Since the contact location is defined over a discrete action space (selecting a contact point among the points in the object point cloud) and the motion parameters

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

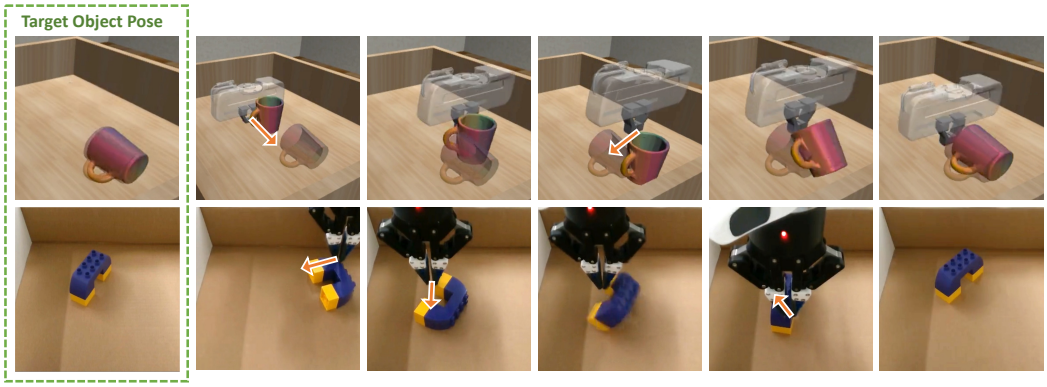


Figure 1: We propose HACMan (**H**ybrid **A**ctor-**C**ritic Maps for **Man**ipulation), which allows non-prehensile manipulation of previously-unseen objects into arbitrary stable poses. With HACMan, the robot learns to push, tilt, and flip the object to reach the target pose, which is shown in the first column and in the top row with transparency. The policy allows for dynamic object motions with complex contact events in both simulation (top) and in the real world (bottom). The performance of the policy is best understood from the videos on the project website: https://hacman-2023.github.io.

(defining the trajectory after contact) are defined over a continuous action space, our action representation is in a hybrid discrete-continuous action space. In HACMan, the actor network outputs *per-point* continuous motion parameters and the critic network predicts *per-point* Q-values over the object point cloud. Different from common continuous action space RL algorithms [6, 7], the per-point Q-values are used both to update the actor and also to compute the probability for selecting the contact location. We modify the update rule of an existing off-policy RL algorithm to incorporate such a hybrid action space.

We apply HACMan to a 6D object pose alignment task with randomized initial object poses, randomized 6D goal poses, and diverse object geometries (Fig. 1). In simulation, our policy generalizes to unseen objects without a performance drop, obtaining an 89% success rate on unseen objects. In addition, HACMan achieves a training success rate more than three times higher than the best baseline with an alternative action representation. We also perform real robot experiments with zero-shot sim2real transfer, in which the learned policy performs dynamic object interactions over unseen objects of diverse shapes with non-planar goals. Our contributions include:

- We propose a novel object-centric action representation based on 3D spatial action maps to learn complex non-prehensile interactions. We also modify an existing off-policy RL algorithm to incorporate such a hybrid discrete-continuous action space.
- The proposed action representation demonstrates substantive improvements of performance over the baselines and shows strong generalization to unseen objects.
- The learned policy showcases complex contact-rich and dynamic manipulation skills, including pushing, tilting, and flipping, shown both in simulation and with a real robot.

2 Related Work

Non-prehensile manipulation. Non-prehensile manipulation is defined as manipulating objects without grasping them [8]. Many non-prehensile manipulation tasks involve complex contact events among the robot, the object, and the environment, which lead to significant challenges in state-estimation, planning and control [1, 2, 9, 5]. Recent work has applied learning-based methods in non-prehensile manipulation, but they are limited in terms of either skill complexity [10, 3, 4, 11] or object generalization [12, 11]. In contrast, our work shows 6D object manipulation involving more complex object interactions while also generalizing to a large variety of unseen object geometries.

Visual Reinforcement Learning with Point Clouds. Recent research has explored various ways of incorporating point clouds into RL [13, 14]. To overcome the optimization difficulties, previous work has tried pre-training the feature extractor with an auxiliary loss [15], initializing the RL policy with behavior cloning [16], or using student-teacher training [17, 18] (see detailed discussion in Appendix E). Our method does not require these additional training procedures due to the benefits of the proposed action representation. In the experiments, we show that the baselines following the most relevant previous work [13, 17, 18] struggles when the task becomes more complex.

Spatial action maps. Similar to our method, recent work has explored spatial action maps that are densely coupled with visual input instead of compressing it into a global embedding, based on images [19, 4, 20], point clouds [21, 3], or voxels [22]. Unlike previous works with spatial action maps that consider one-shot decision making (similar to a bandit problem) [3, 4, 10] or rely on expert demonstrations with ground truth actions [19, 21, 22], our method reasons over multi-step sequences with no expert demonstrations. For example, Xu et al. [4] only chooses a single contact location followed by an action sequence, rather than a sequence of contact interactions. Unlike previous work in spatial action maps that uses DQN with discrete actions [23, 24, 10, 20], our hybrid discrete-continuous action space allows the robot to perform actions without discretization. Furthermore, we demonstrate the benefit of spatial action representations when applied to a 6D non-prehensile manipulation task, which is more challenging than the pick-and-place and articulated object manipulation tasks in previous work.

RL with hybrid discrete-continuous action spaces. Most RL algorithms focus on either a discrete action space [25] or a continuous action space [7, 6, 26]. However, certain applications are defined over a hybrid action space where the agent selects a discrete action and a continuous parameter for the action [27, 28, 29]. Unlike previous work, our hybrid action space uses a spatial action representation in which the discrete actions are defined over a map of visual inputs. The closest to our work is Feldman et al. [20] in terms of applying RL to spatial action maps, but they only consider a finite horizon of 2. We include formal definitions of the policies over the hybrid action space and modify the loss functions and exploration accordingly.

3 Preliminaries

A Markov Decision Process (MDP) models a sequential stochastic process. An MDP can be represented as a tuple (S, A, P, r), where S is the state space; A is the action space; $P(s_{t+1}|s_t, a_t)$ is the transition probability function, which models the probability of reaching state s_{t+1} from state s_t by taking action a_t; $r(s_t, a_t, s_{t+1})$ is the immediate reward at time t. The objective is to maximize the return R_t, defined as the cumulative discounted reward $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$. Given a policy π, the Q-function is defined as $Q^\pi(s, a) = \mathbb{E}_\pi[R_t|s_t = s, a_t = a]$.

HACMan is built on top of Q-learning-based off-policy algorithms [6, 26, 28]. In these algorithms, we define a deterministic policy π_θ parameterized by θ and a Q-function Q_ϕ parameterized by ϕ. Given a dataset D with transitions (s_t, a_t, s_{t+1}), the Q-function loss is defined according to the Bellman residual:

$$L(\phi) = \mathbb{E}_{s_t,a_t,s_{t+1}\sim D}[(Q_\phi(s_t, a_t) - y_t)^2] \tag{1}$$

where y_t is defined as:

$$y_t = r_t + \gamma Q_\phi(s_{t+1}, \pi_\theta(s_{t+1})) \tag{2}$$

The policy loss for π_θ is defined to maximize the Q-function:

$$J(\theta) = -\mathbb{E}_{s_t\sim D}[Q^{\pi_\theta}(s_t, a_t)|_{a_t=\pi_\theta(s_t)}] \tag{3}$$

4 Problem Statement and Assumptions

We focus on the task of 6D object pose alignment with non-prehensile manipulation. The objective for the robot is to perform a sequence of non-prehensile manipulation actions (i.e. pushing, flipping) to move an object on the table into a target goal pose. We assume that the goals are stable object poses on the table. The robot policy observes the point cloud of the scene from depth cameras, denoted as $\mathcal{X}$. We assume that the point cloud observation is segmented between the background and the object to be manipulated. Thus, the full point cloud $\mathcal{X}$ consists of the object point cloud $\mathcal{X}^{obj}$ and the background point cloud $\mathcal{X}^b$.

5 Method

5.1 Action Representation

We propose an object-centric action space that consists of two parts: a **contact location** a^{loc} on an object and **motion parameters** a^m which define how the robot moves to interact with the object after contact. As shown in Fig. 2, to execute an action, the end-effector will first move to a location in free space near location a^{loc}, after which it will interact with the object using the motion parameters a^m. After the interaction, the end-effector will move away from the object, a new observation is obtained, and the next action can be taken.

Specifically, given the object point cloud $\mathcal{X}^{obj} = \{x_i \mid i = 1 \dots N\}$, where $x_i \in \mathbb{R}^3$ are the point locations, the contact location a^{loc} is chosen from among the points in $\mathcal{X}^{obj}$. Thus, a^{loc} is defined over a discrete action space of dimension N. We assume a collision-free motion planner to move the gripper to the contact location a^{loc} (see Appendix A.6 for details). In contrast, the motion parameters a^m, which define how the gripper interacts with the object after contact, are defined in a continuous action space. Furthermore, we define a^m as the end-effector delta position movement from the contact position, hence $a^m \in \mathbb{R}^3$. Our experiments show that translation-only movements are sufficient to enable complex 6D object manipulation in our task. We also include additional experiments on extending motion parameters to enable rotations in Appendix C.3.

Figure 2: **Illustration of our action space.**

The proposed action representation has two benefits compared to previous work. First, it is **temporally-abstracted**. Compared to the common action space of end-effector delta movements [30, 31, 12, 32], the agent with our action space can avoid wasting time learning how to move in free space and instead focus on learning contact-rich interactions. Second, it is **spatially-grounded** since the agent selects a contact location from the observed object point cloud.

5.2 Hybrid RL Algorithm

The proposed action space is a hybrid discrete-continuous action space: the contact location a^{loc} is discrete while the motion parameters a^m are continuous. We propose a way to adapt existing off-policy algorithms designed for continuous action spaces [6, 26] to this hybrid action space.

First, consider the simpler case of an action space that only has the continuous motion parameters a^m. In this case, we can directly apply existing off-policy algorithms as described in Section 3. Given the observation s (which is the point cloud $\mathcal{X}$ in our task), we can train an actor to output a vector of motion parameters $\pi_\theta(s) = a^m$. Similarly, the critic $Q_\phi(s, a^m)$ outputs the Q-value given the observation s and the motion parameters a^m; the Q-value can be used to update the actor according to Eqn. 3.

To additionally predict the contact location a^{loc}, we also need the policy to select a point among the discrete set of points in the object point cloud $\mathcal{X}^{obj}$. Our insight is that we can embed such a discrete component of the action space into the critic by training the critic to output a per-point Q-value Q_i for each point x_i over the entire point cloud. The Q-value at each point on the object represents the estimated return after selecting this point as the contact location. These Q-values can thus be used not only to update the actor, but also to select the contact location as the point with the highest Q-value. Additionally, we train the actor to also output per-point motion parameters a_i^m for each point x_i. If point x_i is selected as the contact location a^{loc}, then the motion parameters at this point a_i^m will be used as the gripper motion after contact.

The overall architecture is shown in Fig. 3. The actor π_θ receives as input the point cloud observation and outputs per-point motion parameters $\pi_\theta(\mathcal{X}) = \{a_i^m = \pi_{\theta,i}(\mathcal{X}) \mid i = 1 \dots N\}$. We call this per-point output an **"Actor Map"**. The critic also receives as input the point cloud observation. It first calculates per-point features $f(\mathcal{X}) = \{f_i \mid i = 1 \dots N\}$. The critic then concatenates each per-point feature f_i with the corresponding per-point motion parameter a_i^m and inputs the concatenated vector to an MLP. The output of the MLP is a per-point Q-value: $Q_i = Q_\phi(f_i, a_i^m)$, which scores the action of moving the gripper to location x_i and executing motion parameters a_i^m. We call this per-point output a **"Critic Map"**. In this way, the critic is able to reason jointly about the contact location (via the feature f_i) as well as the motion parameters a_i^m.

At inference time, we can select the point x_i within the object points $\mathcal{X}^{obj}$ with the highest Q-value Q_i as the contact location and use the corresponding motion parameters a_i^m. For exploration during training, we define a policy π^{loc} which selects the contact location based on a softmax of the Q-values over all of the object points. The probability of a point x_i being selected as the contact location is thus given as:

$$\pi^{loc}(x_i \mid s) = \pi^{loc}(x_i \mid \mathcal{X}) = \frac{\exp(Q_i/\beta)}{\sum_{k=1,\dots,N} \exp(Q_k/\beta)} \tag{4}$$

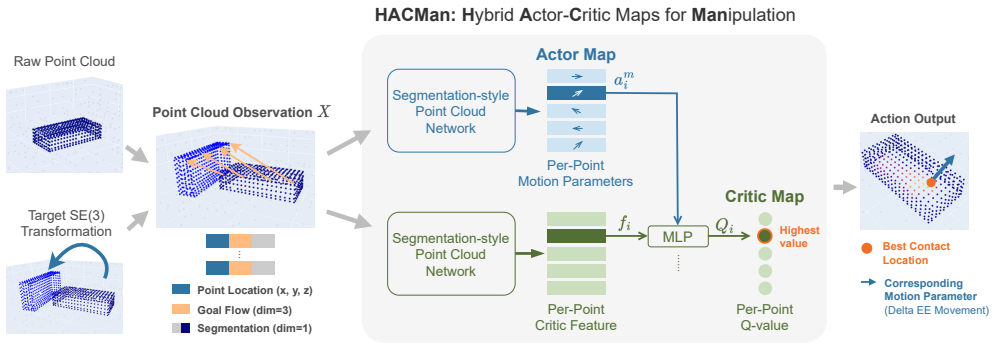


Figure 3: **An overview of the proposed method.** The point cloud observation includes the location of the points and point features. The goal is represented as per-point flow of the object points. The actor takes the observation as input and outputs an **Actor Map** of per-point motion parameters. The Actor Map is concatenated with the per-point critic features to generate the **Critic Map** of per-point Q-values. Finally, we choose the best contact location according to the highest value in the Critic Map and find the corresponding motion parameters in the Actor Map.

where β is the temperature of the softmax, which controls the exploration of the contact location. Note that the background points $\mathcal{X}^b$ are included in the observation $s = \mathcal{X} = \{\mathcal{X}^b, \mathcal{X}^{obj}\}$, but are excluded when choosing the contact location.

We modify the update rules of the off-policy algorithm for this hybrid policy. Given $s = \mathcal{X}$, we first define the per-point loss for updating the actor $\pi_{\theta,i}(s)$ at location x_i according to Eqn. 3:

$$J_i(\theta) = -Q_\phi(f_i, a_i^m) = -Q_\phi(f_i, \pi_{\theta,i}(s)) \tag{5}$$

where f_i is the feature corresponding to point x_i. The total objective of the actor is then computed as an expectation over contact locations:

$$J(\theta) = \mathbb{E}_{x_i \sim \pi^{loc}}[J_i(\theta)] = \sum_i \pi^{loc}(x_i \mid s) \cdot J_i(\theta) \tag{6}$$

where $\pi^{loc}(x_i \mid s)$ is the probability of sampling contact location x_i, defined in Eq. 4. The difference between Eqn. 6 and the regular actor loss in Eqn. 3 is that we use the probability of the discrete action to weight the loss for the continuous action. To take into account π_{loc} during the critic update, the Q-target y_t from Eqn. 2 is modified to be:

$$y = r_t + \gamma \mathbb{E}_{x_i \sim \pi^{loc}}[Q_\phi(f_i(s_{t+1}), \pi_{\theta,i}(s_{t+1}))] \tag{7}$$

5.3 Representing the Goal as Per-Point Goal Flow

As described in Section 4, our task is goal-conditioned object manipulation, in which the objective is to move an object to a given goal pose. Instead of concatenating the goal point cloud to the observed point cloud [17, 18], we represent the goal as per-point "goal flow" of the object point cloud to the goal: Suppose that point x_i in the initial point cloud corresponds to point x_i' in the goal point cloud; then the goal flow is given by $\Delta x_i = x_i' - x_i$. The goal flow Δx_i is a 3D vector which is concatenated to the other features of the point cloud observation. This flow representation of goal is also used to calculate the reward and success rate for the pose alignment task (Appendix A.3). In the real robot experiments, we estimate the goal flow using point cloud registration (Appendix D). Our ablation experiments suggest that utilizing the flow representation of the goal drastically enhances performance compared to directly concatenating the goal point cloud (Appendix C.1).

6 Experiment Setup

We evaluate our method on the 6D object pose alignment task as described in Section 4, in which the objective is for the robot to perform a sequence of non-prehensile manipulation actions (i.e. pushing, flipping) to move the object to a given goal pose. In this section, we describe the task setup in simulation, used for training and simulation evaluation (see Section 8 for real robot experiments).

Task Setup. The simulation environment is built on top of Robosuite [33] with MuJoCo [34]. We include 44 objects with diverse geometries from Liu et al. [35]. Details and visualizations of the object models are included in Appendix A.1. The object dataset is split into three mutually exclusive sets: training set (32 objects), unseen instances (7 objects) and unseen categories (5 objects). The 7 unseen instances consist of objects from categories included in the training set, whereas the 5 objects

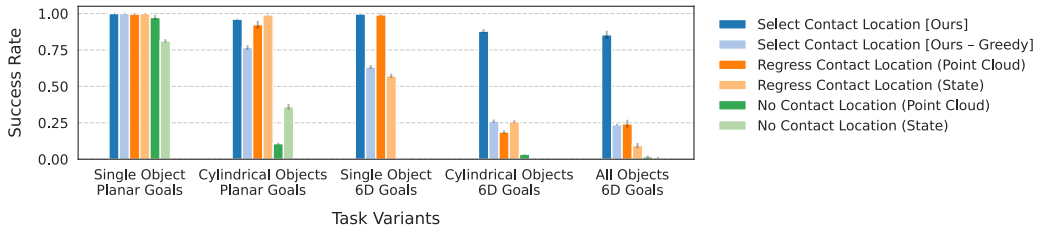


Figure 4: **Baselines and Ablations.** Our approach outperforms the baselines and the ablations, with a larger margin for more challenging tasks on the right. Success rates for simple tasks - pushing a single object to an in-plane goal - are high for all methods, but only HACMan achieves high success rates for 6D alignment of diverse objects.

in "unseen categories" consist of novel object categories. An episode is considered a success if the average distance between the corresponding points of the object and the goal is less than 3 cm. More details on our simulation environment setup can be found in Appendix A.

Table 1: Features of the proposed action representation compared to the baselines.

	Spatially Grounded?	**Temporally Abstracted?**
Select Contact Location [Ours]	✓	✓
Regress Contact Location	×	✓
No Contact Location [17, 18, 12, 13, 31, 30]	×	×

Task Variants. To analyze the limitations of different methods, we design the object pose alignment task with varying levels of difficulty. We consider three types of object datasets: An **All Objects** dataset that includes the full object dataset, a **Cylindrical Objects** dataset consisting of only cylindrical objects, and a **Single Object** dataset consists of just a single cube. We also try different task configurations: In the **Planar goals** experiments, the object starts from a *fixed* initial pose at the center of the bin, and the goal pose is a randomized *planar translation* of the starting pose. In the **6D goals** experiments, both the object initial pose and goal pose are *randomized SE(3)* stable poses, not limited to planar motion. This task requires SE(3) object movement to achieve the goal which imposes challenges in spatial reasoning. These different task variations are used to show at what level of difficulty each of the baseline methods stop being able to complete the task.

7 Simulation Results

In this section, we demonstrate the effectiveness of HACMan compared to the baselines and ablations. Fig. 4 summarizes the performance of each method after being trained with the same number of environment interactions. The training curves, tables, and additional results can be found in Appendix C. Implementation details of all methods are included in Appendix B.

Effect of action representations. We compare our method with two alternative action representations, summarized in Table 1. In **Regress Contact Location**, the policy directly regresses to a contact location and motion parameters, instead of choosing a contact point from the point cloud as in HACMan. The **No Contact Location** baseline directly regresses to a delta end-effector movement at each timestep. For every action, the robot continues from its position from the previous action, instead of first moving the gripper to a selected contact location. This is the most common action space in manipulation [17, 18, 12, 13, 31, 30]. As input for these two baselines, we use either point cloud observations or ground-truth state, establishing four baselines in total. The baseline that regresses motion parameters from point cloud observations is a common action representation used in prior work in RL from point clouds, including state-of-the-art methods such as Qin et al. [13]. The baseline that regresses motion parameters from ground-truth state is the most common approach in prior work, such as in Zhou and Held [12] as well as the teacher policies in Chen et al. [17, 18] (see Appendix E for more discussion).

As shown in Fig. 4, these baseline action representations struggle with the more complex task variants. For the most challenging task variant "All Objects + 6D goals," our method achieves a success rate 61% better than the best baseline (see Table 5 for numbers). As mentioned in Section 5, the proposed action representation benefits from being *spatially-grounded* and *temporally-abstracted*. The comparison against the baselines demonstrates the importance of each of these two features (Table 1). The "Regress Contact Location" baseline still benefits from being temporally-abstracted because the gripper starts from a location chosen by the policy at each timestep; however, this action representation is not spatially-grounded because it regresses to a contact location which might not

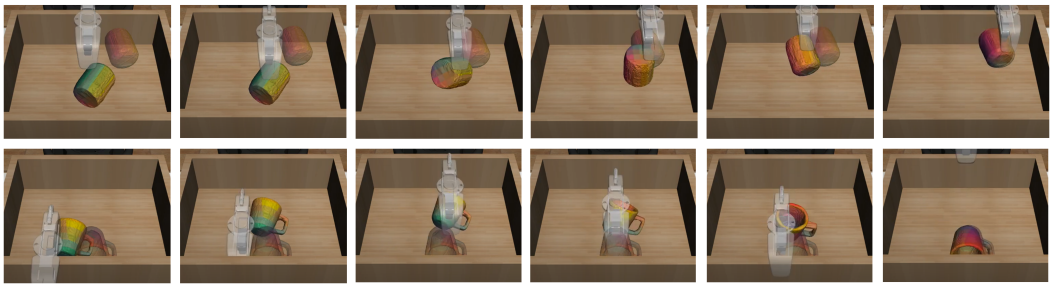


Figure 5: **Qualitative results for the object pose alignment task.** HACMan shows complex non-prehensile behaviors that move the object to the goal pose (shown as the transparent object).

be on the object surface, unlike our approach which selects the contact location among the points in the point cloud observation. Thus, the "Regress Contact Location" baseline suffers from training difficulties with more diverse objects (last two variants in Fig. 4). The "No Contact Location" baseline is neither spatially-grounded nor temporally-abstracted; it follows the usual approach from prior work [17, 18, 12, 13, 31, 30] of regressing an end-effector delta motion at each timestep. While this is the most common action space in prior work, it has close to zero performance with 6D goals.

Effect of Multi-step Reasoning. To test the necessity of multi-step reasoning for the pose alignment task, we experiment with a **"Greedy"** version of HACMan by setting the discount factor γ in the RL algorithm to $\gamma = 0$. This forces the algorithm to optimize for greedy actions for each step. Using RL for multi-step reasoning is one of the important differences between our method and previous work such as Where2Act [3] and UMPNet [4] which optimize for one-step contact locations. Fig. 4 indicates that greedy actions might work for planar goals, but suffer from poor performance for 6D goals that requires multi-step non-greedy interactions. For example, the last row in Fig. 5 shows an example of our method pushing the object away from the goal position to prepare for flipping it to the correct orientation, demonstrating non-greedy behavior. In contrast, we find that the greedy ablation often results in local optima of only trying to match the object position but not its orientation.

Generalization to unseen objects. The evaluation of our method over unseen objects with 6D goals is summarized in Table 2. Our method generalizes well to unseen object instances and unseen categories without a performance drop. When we increase the maximum episode length from 10 steps to 30 steps, our method achieves 95.1% success on unseen categories (Appendix C.6). Table 2 shows that, comparing the same set of object categories ("common categories"), the success rates of the training instances are similar to the unseen instances. The differences in geometry comparing the training objects and the unseen objects are visualized in Appendix A.1.

Table 2: **Generalization to unseen objects.**

Object Set Split	Success Rate	# of Objects
Train	0.833 ± .018	32
Train (Common Categories)	0.887 ± .024	13
Unseen Instance (Common Categories)	0.891 ± .033	7
Unseen Category	0.827 ± .047	5

Goal-conditioned Object Affordance and Multimodality. We visualize the Critic Map, which computes the score of each contact location of the object (Fig. 6). The Critic Maps capture goal-conditioned object affordances which describe how the object can be moved to achieve the goal. Fig. 6(a) and (b) are two scenarios of performing translation object motions for different object heights: For a thin object, the Critic Map highlights the region of the top of the object (dragging) or from the back side of the object (pushing). For a thick object, it prefers to push from the bottom to avoid the object falling over. In Fig. 6(c), the hammer needs to be rotated by 180 degrees. The Critic Map predicts a multimodal solution of pushing from ei-

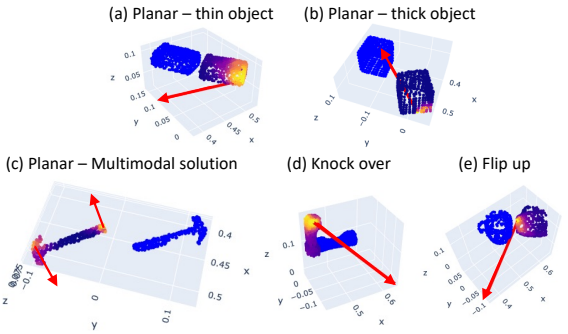
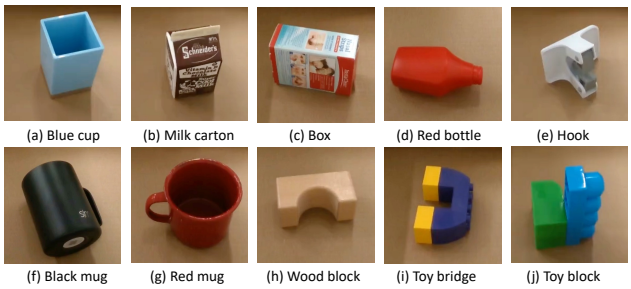


Figure 6: **Visualizations of Goal-conditioned Critic Maps.** Blue: goal point cloud. Color map: observed object point cloud. Lighter colors indicate higher Critic Map scores. Red arrows: motion parameters at a selected contact location. The policy uses different contact locations based on object geometries and goals.



Object Name	Planar Goals	Non-planar Goals	Total
(a) Blue cup	4/7	4/13	8/20
(b) Milk bottle	6/7	10/13	16/20
(c) Box	2/5	10/15	12/20
(d) Red bottle	4/7	0/13	4/20
(e) Hook	5/8	5/12	10/20
(f) Black mug	4/7	0/13	4/20
(g) Red mug	5/7	3/13	8/20
(h) Wood block	6/7	6/13	12/20
(i) Toy bridge	9/10	5/10	14/20
(j) Toy block	2/2	10/18	12/20
Total	47/67	53/133	100/200
Percentage	70%	40%	50%

Figure 7: **Real robot experiments.** HACMan achieves a 50% overall success rate over unseen objects with different geometries and physical properties, with 6D goal poses.

ther end of the object. Fig. 6(d) and (e) show out-of-plane motions of the object of knocking over and flipping up the objects.

Additional Results. We include additional experiment results in Appendix C, including additional ablations in Appendix C.1, extending the motion parameters in Appendix C.3, cluttered scenes in Appendix C.4, longer training steps in Appendix C.5, longer episode lengths in Appendix C.6, and success rate breakdown for each object category in Appendix C.7.

8 Real robot experiments

In the real robot experiments, we aim to evaluate the ability of the trained policy to generalize to novel objects and execute dynamic motions in the real world. We evaluate the policy with a diverse set of objects with different shapes, surface frictions, and densities (Fig. 7). We use random initial poses and random 6D goals (referred to in the previous section as "All Objects + 6D Goals"). For example, the red mug (Fig. 7(g)) has goal poses of being upright on the table or lying on the side. An episode is considered a success if the average distance of corresponding points between the object and the goal is smaller than 3 cm; we also mark an episode as a failure if there is a failure in the point cloud registration between the observation and the goal. Implementation details of the real robot experiments can be found in Appendix D.

Fig. 7 (Right) summarizes the quantitative results of the real robot experiments. We run the evaluations without manual reset, which may create uneven numbers of planar versus out-of-plane goals. It achieves 70% success rate on planar goals and 40% success rate on non-planar goals. Non-planar goals are more difficult than the planar goals because they require dynamic motions to interact with the object. A small error in the action may result in large changes in the object movement. Videos of the real robot experiments can be found on the website: `https://hacman-2023.github.io`. The real robot experiments demonstrate that the policy is able to generalize to novel objects in the real world, despite the sim2real gap of the simulator physics and inaccuracies of point cloud registration for estimating the goal transformation. More discussion can be found in Appendix D.

9 Limitations

Since the contact location in our action space is defined over the point cloud observation, our method requires relatively accurate depth readings and camera calibration. Further, the contact location is currently limited to the observed part of the object. In addition, for this goal-conditioned task, we represent the goal as per-point flow (Section 5.3) which relies on point cloud registration algorithms. Inaccuracies in the registration algorithm sometimes leads to failure cases in real robot experiments.

10 Conclusion

In this work, we propose to learn Hybrid Actor-Critic Maps with reinforcement learning for non-prehensile manipulation. The learned policy shows complex object interactions and strong generalization across unseen object categories. Our method achieves a significantly higher success rate than alternative action representations, with a larger performance gap for more difficult task variants. The proposed method has the potential to be extended and used in a wider variety of manipulation tasks. We hope the proposed method and the experimental results can pave the way for future work on more skillful robot manipulation over a diverse set of objects.

References

[1] X. Cheng, E. Huang, Y. Hou, and M. T. Mason. Contact mode guided motion planning for quasidynamic dexterous manipulation in 3d. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2730–2736. IEEE, 2022.

[2] Y. Hou and M. T. Mason. Robust execution of contact-rich motion plans by hybrid force-velocity control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1933–1939. IEEE, 2019.

[3] K. Mo, L. J. Guibas, M. Mukadam, A. Gupta, and S. Tulsiani. Where2act: From pixels to actions for articulated 3d objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6813–6823, October 2021.

[4] Z. Xu, H. Zhanpeng, and S. Song. Umpnet: Universal manipulation policy network for articulated objects. *IEEE Robotics and Automation Letters*, 2022.

[5] K.-T. Yu, M. Bauza, N. Fazeli, and A. Rodriguez. More than a million ways to be pushed. a high-fidelity experimental dataset of planar pushing. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 30–37, 2016. doi: 10.1109/IROS.2016.7758091.

[6] S. Fujimoto, H. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[7] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[8] M. T. Mason. Progress in nonprehensile manipulation. *The International Journal of Robotics Research*, 18(11):1129–1141, 1999.

[9] I. Mordatch, E. Todorov, and Z. Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics (TOG)*, 31(4):1–8, 2012.

[10] J. Wu, X. Sun, A. Zeng, S. Song, J. Lee, S. Rusinkiewicz, and T. Funkhouser. Spatial action maps for mobile manipulation. In *Proceedings of Robotics: Science and Systems (RSS)*, 2020.

[11] J. Liang, X. Cheng, and O. Kroemer. Learning preconditions of hybrid force-velocity controllers for contact-rich manipulation. *arXiv preprint arXiv:2206.12728*, 2022.

[12] W. Zhou and D. Held. Learning to grasp the ungraspable with emergent extrinsic dexterity. In *Conference on Robot Learning (CoRL)*, 2022.

[13] Y. Qin, B. Huang, Z.-H. Yin, H. Su, and X. Wang. Dexpoint: Generalizable point cloud reinforcement learning for sim-to-real dexterous manipulation. *Conference on Robot Learning (CoRL)*, 2022.

[14] M. Liu, X. Li, Z. Ling, Y. Li, and H. Su. Frame mining: a free lunch for learning robotic manipulation from 3d point clouds. In *6th Annual Conference on Robot Learning*, 2022.

[15] W. Huang, I. Mordatch, P. Abbeel, and D. Pathak. Generalization in dexterous manipulation via geometry-aware multi-task learning. *arXiv preprint arXiv:2111.03062*, 2021.

[16] L. Wang, Y. Xiang, W. Yang, A. Mousavian, and D. Fox. Goal-auxiliary actor-critic for 6d robotic grasping with point clouds. In *The Conference on Robot Learning (CoRL)*, 2021.

[17] T. Chen, J. Xu, and P. Agrawal. A system for general in-hand object re-orientation. *Conference on Robot Learning*, 2021.

[18] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal. Visual dexterity: In-hand dexterous manipulation from depth. *arXiv preprint arXiv:2211.11744*, 2022.

[19] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani, and J. Lee. Transporter networks: Rearranging the visual world for robotic manipulation. *Conference on Robot Learning (CoRL)*, 2020.

[20] Z. Feldman, H. Ziesche, N. A. Vien, and D. Di Castro. A hybrid approach for learning to shift and grasp with elaborate motion primitives. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6365–6371. IEEE, 2022.

[21] D. Seita, Y. Wang, S. Shetty, E. Li, Z. Erickson, and D. Held. ToolFlowNet: Robotic Manipulation with Tools via Predicting Tool Flow from Point Clouds. In *Conference on Robot Learning (CoRL)*, 2022.

[22] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Proceedings of the 6th Conference on Robot Learning (CoRL)*, 2022.

[23] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4238–4245. IEEE, 2018.

[24] A. Hundt, B. Killeen, N. Greene, H. Wu, H. Kwon, C. Paxton, and G. D. Hager. "good robot!": Efficient reinforcement learning for multi-step visual tasks with sim to real transfer. *IEEE Robotics and Automation Letters*, 5(4):6724–6731, 2020.

[25] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[26] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[27] M. Neunert, A. Abdolmaleki, M. Wulfmeier, T. Lampe, T. Springenberg, R. Hafner, F. Romano, J. Buchli, N. Heess, and M. Riedmiller. Continuous-discrete reinforcement learning for hybrid control in robotics. In *Conference on Robot Learning*, pages 735–751. PMLR, 2020.

[28] M. Hausknecht and P. Stone. Deep reinforcement learning in parameterized action space. *arXiv preprint arXiv:1511.04143*, 2015.

[29] J. Xiong, Q. Wang, Z. Yang, P. Sun, L. Han, Y. Zheng, H. Fu, T. Zhang, J. Liu, and H. Liu. Parametrized deep q-networks learning: Reinforcement learning with discrete-continuous hybrid action space. *arXiv preprint arXiv:1810.06394*, 2018.

[30] T. Mu, Z. Ling, F. Xiang, D. C. Yang, X. Li, S. Tao, Z. Huang, Z. Jia, and H. Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[31] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019. URL https://arxiv.org/abs/1910.10897.

[32] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016.

[33] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany, and Y. Zhu. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020.

[34] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. doi:10.1109/IROS.2012.6386109.

[35] W. Liu, T. Hermans, S. Chernova, and C. Paxton. Structdiffusion: Object-centric diffusion for semantic rearrangement of novel objects. *arXiv preprint arXiv:2211.04604*, 2022.

[36] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Computer Vision–ACCV 2012: 11th Asian Conference on Computer Vision, Daejeon, Korea, November 5-9, 2012, Revised Selected Papers, Part I 11*, pages 548–562. Springer, 2013.

[37] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *Robotics: Science and Systems (RSS)*, 2018.