DexPoint: Generalizable Point Cloud Reinforcement Learning for Sim-to-Real Dexterous Manipulation

Yuzhe Qin*[1] Binghao Huang*[1] Zhao-Heng Yin[2]
Hao Su[1] Xiaolong Wang[1]

UC San Diego[1] HKUST[2]

Abstract: We propose a sim-to-real framework for dexterous manipulation which can generalize to new objects of the same category in the real world. The key of our framework is to train the manipulation policy with point cloud inputs and dexterous hands. We propose two new techniques to enable joint learning on multiple objects and sim-to-real generalization: (i) using imagined hand point clouds as augmented inputs; and (ii) designing novel contact-based rewards. We empirically evaluate our method using an Allegro Hand to grasp novel objects in both simulation and real world. To the best of our knowledge, this is the first policy learning-based framework that achieves such generalization results with dexterous hands. Our project page is available at https://yzqin.github.io/dexpoint.

Keywords: Dexterous Manipulation, Point Clouds, Sim-to-Real

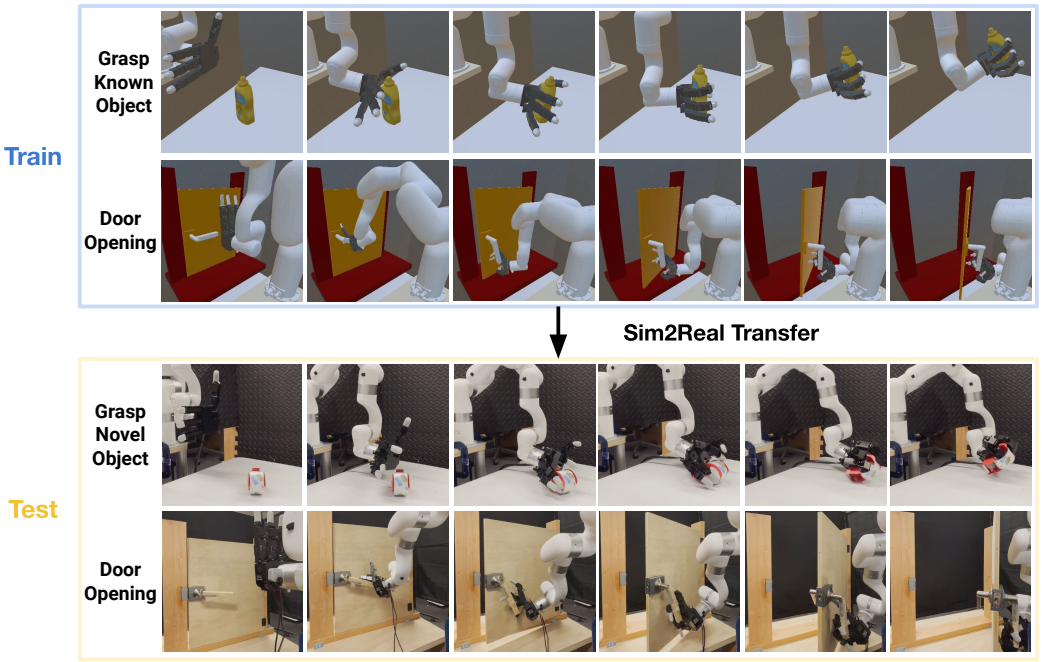


Figure 1: We introduce a reinforcement learning method which takes the point cloud as input for two manipulation tasks: grasping and door opening. By introducing several techniques in the policy learning process, our point cloud based policy trained purely in simulation can successfully generalize to novel objects and transfer to real-world without any real-world data.

* indicates equal contribution.

6th Conference on Robot Learning (CoRL 2022), Auckland, New Zealand.

1 Introduction

Dexterous manipulation has remained to be one of the most challenging problems in robotics [1]. While multi-finger hands create ample opportunities for robots to flexibly manipulate objects in our daily life, the nature of the high degree of freedom and high-dimensional action space creates significant optimization challenges for both search-based planning algorithms and policy learning algorithms. Recent efforts using model-free Reinforcement Learning have achieved encouraging results on complex manipulation tasks [1, 2]. However, it still faces many challenges in generalizing to diverse objects and being deployed on multi-finger hand in the real-world.

For example, the dexterous manipulation framework proposed by OpenAI et al. [1] can solve in-hand manipulation of Rubik's Cube with RL and transfer to the real robot hand. However, the policy is only trained with one particular object and it is not able to *generalize to diverse objects*. To achieve cross-object generalization, recent efforts proposed to learn robust 3D point cloud representations [3, 4, 5, 6] with diverse objects using RL in simulation. While point cloud input has also been shown easier for Sim2Real transfer [7] given its focus on geometry instead of texture, the assumption on the access of complete object point clouds and ground truth states limit the transferability of above methods to the *real robot deployment*. Among these works, Chen et al. [3] showed that cross-object generalization is achievable in simulation without knowing the shape of the object to grasp, but its requirement of real-time access to the object states is itself a very challenging robot perception problem, especially under large occlusions during hand object interaction.

In this paper, we provide a sim-to-real reinforcement learning framework for generalizable dexterous manipulation, using two tasks with the Allegro Hand[8]: (i) object grasping where the test objects has not been seen during training; (ii) door opening where the test doors have levers of novel shape that has not been used in training. The tasks are visualized in Figure 1.

the robot needs to first rotate the lever to unlock the door latch and then pull the lever in a circular motion to open the door.

We perform our studies by training a point cloud based reinforcement learning policy in the grasping and door opening task. With this approach, we list the three key discoveries of our framework for learning generalizable point cloud policy below:

(i) We justified that it is possible to achieve *direct sim-to-real transfer* for a dexterous manipulation policy with category-level generalizability when we use point cloud as the data representation.

(ii) Raw point clouds captured by sensors often come with heavy occlusions and noise: only a very small portion of the points from the observation are representing the robot fingers. We propose to *imagine* the complete robot finger point clouds according to the robot kinematic model and use them to augment the occluded real point cloud observations. We find that explicitly augmenting the input by *imagined* points can help achieve better robustness and sample efficiency for reinforcement learning.

(iii) Different from existing works that add contact information to the input of RL, we design a novel reward using contact pair information without adding contact to the observation. This practice remarkably improves sample efficiency as well as learning stability and avoids the dependency on contact sensor that is often unavailable for real robot models.

2 Related Work

Dexterous Manipulation. Dexterous manipulation aims to enable robotic hands to achieve human-level dexterity for grasping and manipulating objects. Analytical methods have been proposed to adopt planning to solve this problem [9, 10, 11, 12, 13, 14, 15]. However, they rely on detailed object models, which are not accessible when testing on unseen objects. To mitigate this issue, researchers propose a learning-based method for dexterous manipulation [16]. Some methods [17, 1, 2, 3, 5] take a reinforcement learning (RL) approach, while another line of work [18, 19, 20, 21, 22] also proposes to learn from demonstration using imitation learning (IL) to acquire the control policy. Recently, the combination of RL and IL [23, 24, 25, 26, 27, 4] has also shown encouraging dexterous manipulation results in simulation. However, most methods are either learning policy for one single object or assume access to object states produced by perfect detector which increases challenges to

Sim2Real transfer. In this paper, we surpass these limitations by introducing training on multiple objects and applying point cloud inputs for control.

Point Cloud in Robotic Manipulation. Point cloud representation has been widely applied in the robotics community. Researchers have studied matching the observed point clouds to an object in a grasping dataset [28] and executing the corresponding grasping action [29, 14]. For learning-based approaches, one line of research has focused on first estimating grasp proposals or affordance given the point cloud input and then planning accordingly for manipulation [30, 31, 32, 33, 34, 35, 36, 37, 38]. While these methods are designed for parallel-jaw grippers, recent advancements have also been made for grasping with dexterous hands with similar approaches [39, 40, 41, 42]. However, this line of methods requires feedback from motion planning to estimate whether a grasp is plausible. Oftentimes, a stable grasp pose is proposed but it is not achievable by planning. To achieve efficient and flexible manipulation, a Reinforcement Learning policy with point cloud inputs is proposed [3, 5, 4], which allows the robot hand to flexibly adjust its pose while interacting with the object. However, these approaches still face challenges in transferring to the real robot, given the noisy, occluded point clouds in the real world, and training RL policy with noisy point clouds increases the optimization difficulty. In this paper, we propose to use imagined point clouds, contact information, and multi-object training to combat these problems and achieve Sim2Real generalization.

Using Contact Information for Manipulation. Humans can manipulate objects purely from tactile sensing without seeing them. This biological fact inspires researchers to integrate contact and tactile information into the learning pipeline [43, 44, 45, 46, 47, 48, 49]. For example, both tactile and visual information inputs are combined together in [48] for decision making. The tactile information is also utilized as inputs for RL-based manipulation policies [50, 51, 52, 24, 53]. For example, visual-tactile sensor is used with an Allegro hand in simulator for playing piano [53]. Instead of using contacts as inputs, some methods also use contact and tactile information as reward to encourage exploration and boost policy learning [51, 24]. Motivated by these works, we provide a novel design of reward based on each contact link of the robot hand, which encourages more reasonable grasping behavior. Our design does not require a real tactile sensor when training in simulation or deploying in the real robot.

3 Approach

Our objective is to train a generalizable point cloud policy on a dexterous robot hand-arm system that is able to grasp a wide range of objects or open an closed door with RL. We aims at Sim-to-Real transfer without any real-world training or data. During testing, the robot can only access the single-viewed point cloud and the robot proprioception data. As is discussed before, training such a policy comes with numerous technical challenges, including reward design and imperfect point cloud information. In this work, we propose a novel reward design technique based on contact and imagined point cloud model to deal with these challenges.

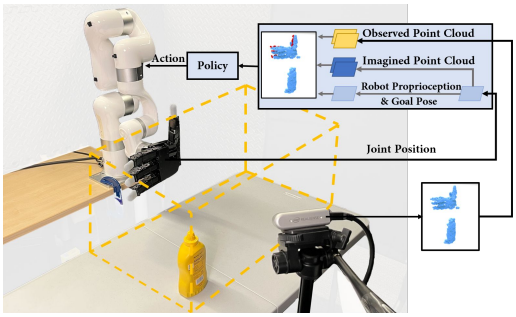


Figure 2: **Real-experiment Setup:** we use an Allegro Hand attached on an XArm6 and a RealSense D435 camera facing forward the robot.

Preliminaries: We model the dexterous manipulation problem as a Partially Observable Markov Decision Process (POMDP) $\mathcal{M} = (\mathcal{O}, \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \mathcal{U})$. Here, $\mathcal{O}$ is the observation space, $\mathcal{S}$ is the underlying state space, $\mathcal{A}$ is the action space, $\mathcal{R}$ is the reward function, $\mathcal{T}$ is the transition dynamics, and $\mathcal{U}$ generates agent's observation. At timestep t, the environment is at the state $s_t \in \mathcal{S}$. The agent observes $o_t \sim \mathcal{U}(\cdot|s_t) \in \mathcal{O}$. The agent takes action a_t and receives reward $r_t = \mathcal{R}(s_t, a_t)$. The environment state at timestep $t+1$ then transit to $s_{t+1} \sim \mathcal{T}(s_t, a_t)$. The objective of the agent is to maximize the return $\sum_{t=0}^{T} \gamma^t r_t$, where γ is a discount factor.

System Setup: Many previous works on learning-based dexterous manipulation attach the hand to a fixed platform to simplify the experiment environment in the real world. In this work, we create a more flexible and powerful dexterous manipulation system which includes both the robotic hand and the arm (Figure 2). Concretely, we attach the Allegro Hand to an XArm6 robot. Allegro Hand

is a 16-DoF anthropomorphic hand with four fingers and XArm6 is a 6-DoF robot arm. We place a RealSense D435 camera at the right front of the robot to capture the point cloud. This setup brings additional challenges to RL exploration and Sim2Real deployment. We use SAPIEN [54] platform with uses a full-physics simulator to build the environment of the whole system. The simulation time step is $0.005s$ to ensure stable contact simulation. Each control step lasts for $0.05s$.

Tasks and Objects: In this paper, we expect our robot to perform the grasping task over a diverse set of objects and to open a locked door by rotating the lever. In the grasping task, we first select a random object from an object dataset and place it on the table. The robot is then required to move it to a target pose. Moreover, the robot should be able to generalize to different initial states, so we randomize both the initial pose and the goal pose for each trial. In simulation experiments, we use bottles and cans from both ShapeNet [55] dataset and YCB [56] dataset. In real-world experiments, we use novel unseen objects to test the policy. In the door opening task, the robot hand is required first to rotate the lever to unlock the door latch and then pull the lever in a circular motion. We use three doors to test the policies both in the simulation and the real world, only one is used for training and the other two are unseen doors. We also randomize the initial pose for each trial.

Observation Space: The observation contains both visual and proprioceptive information with four modalities: (1) Observed point cloud provided by the camera; (2) Proprioception signals of the robot including joint positions and end-effector position; (3) Imagined hand point cloud proposed in Sec. 3.2; (4) Object goal position provided in each trial. All the information is accessible on the real robot. The dimension of each observation modality is shown in Figure 3.

Action Space: The action is responsible for controlling both the 6-DOF robot arm and the 16-DOF hand. It has $6 + 16 = 22$ dimension in total. The robot arm is parametrized by the 6D translation and rotation of the end-effector relative to a reference pose. We use the damped least square inverse kinematics solver with a damping constant $\lambda = 0.05$ to compute the joint motion. Each finger joint of the Allegro hand is controlled by a position controller. Both robot arm and hand are controlled by PD controllers.

Network Architecture: The network architecture is visualized in Figure 3,

3.1 Reward Design with Oracle Contact

Since we aim to solve the dexterous manipulation problem with pure RL, the reward design is central to the method. We need a good reward function to ensure proper interaction between the robotic hand and the object. The whole interaction process consists of two phases. The first phase is to simply reach the object. The second phase is to grasp the object and move it to the target, which is more challenging. For the first phase, we encourage reaching with the following reaching reward:

$$r_{\text{reach}} = \sum_{\text{finger}} \frac{1}{\epsilon_r + d(\mathbf{x}_{\text{finger}}, \mathbf{x}_{\text{obj}})}. \tag{1}$$

Here, $\mathbf{x}_{\text{finger}}$ and $\mathbf{x}_{\text{obj}}$ are the Cartesian position of each fingertip and the target object. Note that $\mathbf{x}_{\text{obj}}$ is available when we perform training in simulation. However, using this reward alone cannot ensure proper contact for grasping. For example, the robot can touch the object with the back of the hand rather than the palm and then get stuck in this local minimum. Therefore, we introduce a novel contact reward to guarantee meaningful contact behavior:

$$r_{\text{contact}} = \textbf{IsContact}(\text{thumb}, \text{object}) \textbf{ AND } \left(\sum_{\text{finger}} \textbf{IsContact}(\text{finger}, \text{object}) \geq 2 \right). \tag{2}$$

This contact reward function outputs a boolean value in $\{0, 1\}$. It outputs 1 only if the thumb is in contact with the object and there are more than one finger in contact with the object. Intuitively, it encourages the robot to cage the object within fingers. In this case, the robot can quickly find out stable grasping and lift the object to the target location. The lifting behavior is encouraged by

$$r_{\text{lift}} = r_{\text{contact}} \textbf{Lift}(\mathbf{x}_{\text{obj}}, \mathbf{x}_{\text{target}}). \tag{3}$$

The **Lift** function is basically in the form of Equation 1 and the main difference is that it will return a large reward value upon task completion. The overall reward function is a weighted combination of the terms above plus a control penalty:

$$\mathcal{R} = w_{\text{reach}} r_{\text{reach}} + w_{\text{contact}} r_{\text{contact}} + w_{\text{lift}} r_{\text{lift}} + w_{\text{penalty}} r_{\text{penalty}}. \tag{4}$$

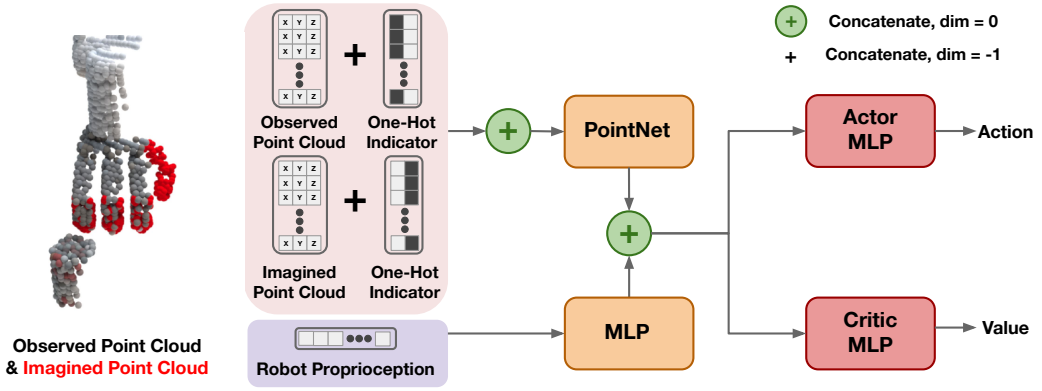


Figure 3: **Architecture:** our feature extractor takes the observed point cloud, imagined point cloud, robot proprioception, and goal pose as input to output a feature embedding. Both actor and critic take the same feature to predict action and value. The red point represented the imaged point cloud of robot hand. Note that our network does not require RGB information.

3.2 Imagined Hand Point Cloud

The usage of point cloud comes with two challenges. The first challenge is occlusion, which may occur to both the object under manipulation and the hand itself. When the robot hand is interacting with an object, the fingers may be occluded by the object. Since we do not assume tactile sensors in this work, this occlusion problem can be serious. The second challenge is the low point cloud resolution during RL training, where we can only use a limited number of points due to the memory limit. In this case, the number of points from hand finger may not be adequate to precisely capture the spatial relationship between the robot and the object. We propose a simple yet effective method to handle both issues in a unified manner. Our idea is to use an imagined hand point cloud in the observation to help the robot to *see the interaction*.

We provide one example in Figure 3. Black points indicate the point cloud captured by the camera, in which some important details of fingers are missing. These missing details provide crucial information of the interaction. Though such interaction information can also be inferred by combining the information from both the proprioception and visual input, we find that the best way is to synthesize these missing details. Concretely, we can compute the pose for each finger link via forward kinematics given the joint position from the robot joint encoder and the robot kinematics model. Then, we synthesize the imagined point cloud (blue points in Figure 3) by sampling the points from the mesh of each finger link. This process is possible in both simulation and the real world.

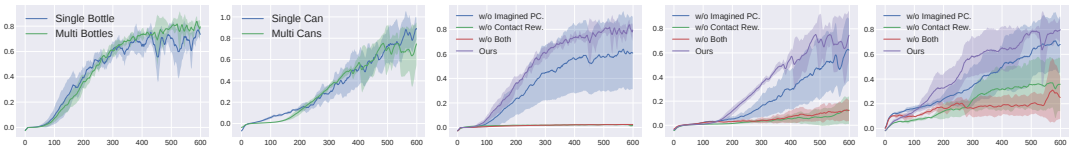
3.3 Training

We adopt Proximal Policy Optimization (PPO) [57] to train the agent in simulation, and then deployed to real without real-world fine-tuning. The network architecture is illustrated in Figure 3. Both value and policy networks share the same visual feature extraction backbone. We concatenate the observed point cloud with the imagined hand point cloud together as the input to the feature extractor. We also attach a one-hot encoding to each point which indicates whether it is observed or an imagined point.

4 Experiments

4.1 Experimental Setup

Point Cloud Pre-processing: To enable smooth transfer from simulation to real-world, we apply the same data preprocessing procedure to the point cloud captured by the camera. It involves four steps: (i) Crop the point cloud to the work region with a manually-defined bounding-box; (ii) Downsample the point cloud uniformly to 512 points; (iii) Add a distance-dependent Gaussian noise to the simulated point cloud to improve the sim2real robustness; (iv) Transform point cloud from the camera frame to the robot base frame using camera pose. In simulation, we use the ground-truth



(a) (b) (c) (d) (e)

Figure 4: **Training Curves.** The left two plots show the single-object and multi-object training curve of (a) bottle category and (b) can category. The right three plots show the ablation results on the (c) grasping bottle (d) grasping can and (e) door opening. The x-axis is the training iterations and y-axis is the normalized episodic return. The shaded area indicates standard error and the performance is evaluated on five random seeds.

Settings	Bottle		Can	
	Known Obj.	Novel Obj.	Known Obj.	Novel Obj.
Single Obj. Training	0.81 ± 0.09	0.60 ± 0.06	0.96 ± 0.04	0.63 ± 0.18
Multi Obj. Training	$\mathbf{0.83 \pm 0.16}$	$\mathbf{0.81 \pm 0.15}$	$\mathbf{0.93 \pm 0.07}$	$\mathbf{0.68 \pm 0.09}$

Table 1: **Experiment on Multi-object Training.** We evaluate the policy trained with single and multiple objects on bottle (upper two rows) and can (bottom two rows) categories with point cloud input. We test them on both known or novel objects. The success rate are reported on 5 seeds.

camera pose with multiplicative noise for frame transformation. In the real-world, we perform hand-eye calibration to get the camera extrinsic parameters.

Evaluation Criterion: We evaluate the performance of a policy by its success rate. For grasping tasks, a task is considered a success if $d_{ot} < 0.05m$ in simulation, where d is the distance between object position and goal position. In real world, the task is considered as success if the XY position of the object is within 5cm from the target position and the height of the object is at least 15cm from table top. For the door opening, the task is considered as success if the door is opened to at least around 45 degrees.

EigenGrasp Baseline: We choose the EigenGrasp [58] as the grasp representation. Given an object mesh model, we use the GraspIt [59] to search valid grasp for Allegro Hand. Then, we use the RRTConnect [60] motion planner implemented in OMPL [61] to plan a joint trajectory to the pre-grasp pose and then plan a screw motion from pre-grasp pose to the grasp pose. Finally, we close all fingers based on searched grasp pose and lift the object to the target. Note that different from our approach, the baseline method **requires complete object model to search for grasps and ground-truth object pose** to align the grasp pose in the robot frame. To evaluate the performance of baseline on novel objects, we first build a grasp database on ShapeNet bottle and can categories using GraspIt. Given the sensory data of a new object, we search for the most similar objects in the dataset and use the query grasps for the novel object. Here we compare the performance of our method with baselines in the real-world.

Training: We train RL in two settings for grasping: (i) training on a single object (ii) training on multiple objects jointly. For the single object grasping, we perform experiments on both "tomato soup can" and "mustard bottle" from YCB. For the multi-object training, we choose 10 objects from the can or bottle categories of ShapetNet. We randomly choose one object to train for each episode in the multi-object training. Here, we use can and bottle as experimental subjects since they represent two different basic grasping patterns [62] for anthropomorphic hand: precision grasp and power grasp. For door opening, we only train the policy on the door with fixed lever geometry.

4.2 Comparison of Single-object and Multi-object Training

We plot the training curve of RL in Figure 4 (a) and (b). In general, our method can learn to grasp and move the object to the goal pose within 600 iterations, where each iteration contains 20K environment steps. Then we evaluate the policy trained on both known and novel objects, and the results are shown in Table 1. We run 100 trials to compute the average success rate. On grasping known objects, we find that agent trained on single-object outperforms the agent trained on multi-object by a small margin, for both learning efficiency and final success rate. However, agent trained on multi-objects does much better at grasping novel objects. Our results suggest that using multiple object during training is important, and is of great importance for novel object generalization.

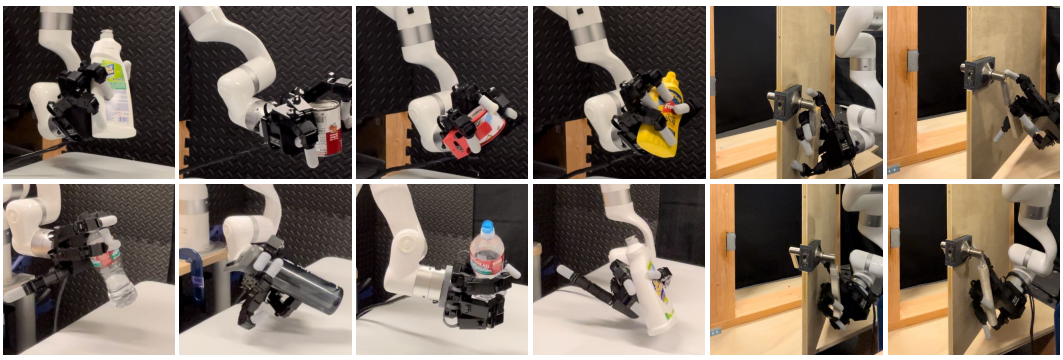


Figure 5: **Real-experiment:** We evaluate our point cloud policy on various unseen objects.

Settings	Bottle		Can		Door	
	Known Obj.	Novel Obj.	Known Obj.	Novel Obj.	Known Obj.	Novel Obj
w/o Imagined PC.	0.60 ± 0.46	0.56 ± 0.51	0.91 ± 0.17	0.63 ± 0.07	0.14 ± 0.28	0.11 ± 0.27
w/o Contact Rew.	0.00 ± 0.00	0.00 ± 0.00	0.06 ± 0.08	0.03 ± 0.06	0.21 ± 0.26	0.20 ± 0.25
w/o Both	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Ours	$\mathbf{0.83 \pm 0.16}$	$\mathbf{0.81 \pm 0.15}$	$\mathbf{0.93 \pm 0.07}$	$\mathbf{0.68 \pm 0.09}$	$\mathbf{0.92 \pm 0.06}$	$\mathbf{0.79 \pm 0.11}$

Table 2: **Ablation Study:** we investigate the influence of contact-based reward design and imaged point cloud. We evaluate the success rate on both known and novel objects under four settings: (i) without imaged point cloud; (ii) without contact reward; (iii) without both; (iv) with both.

4.3 Ablation Results in Simulation

We ablate two key innovations of the work: the reward design with oracle contact and the imaged hand point cloud. We perform experiments on four different variants: (i) without imagined point cloud; (ii) without contact-based reward design; (iii) without both imagined point cloud and contact based reward design; (iv) our standard approach with both techniques. Note that the variant (iii) is an approximation of [3] in our environments and tasks. We compare both the learning curve and the evaluation success rate of these four variants. For grasping task, Figure 4 (c) and (d) show the results on bottle and can categories, and Table 2 shows the success rate. Our findings can be summarized as follows.

First, we find that contact reward information is of vital importance for training the point cloud RL policy on the multi-finger robot hand. Without using contact reward, the agent can hardly learn anything (red and green curve in the figure) and get nearly zero success rate during evaluation for both bottle and can categories. By encouraging the contact between fingers and object, the RL agent can avoid getting stuck in local minimums and learn meaningful manipulation behavior.

Second, the imagined point cloud can also improve the training and test performance for both categories, though it is not so important as the contact reward. As is shown in the learning curves of Figure 4 (c) and (d), the policy utilizing the imagined point cloud as input can learn faster in the early stage of the training and show much smaller variances. An interesting fact is that imagined point cloud is more beneficial for the bottle category than the can category. One possible reason is that grasping a bottle requires multi-finger coordination to perform a power grasp. Such coordination is highly dependent on the detailed finger information provided by the imagined point cloud. The imagined point cloud can help the agent to better see the fingers even if they are occluded by the object, e.g., fingers behind the object.

The experiments on the door opening task also support these findings. As shown in Figure 4 (e), the policy trained with contact based reward and imaged hand point cloud outperform other ablation methods, which also demonstrate the effectiveness of our two key design. Compared with the grasping experiments, we can observe larger variations during policy training. One possible reason is that door opening suffers from heavier occlusion than grasping, when the door lever is grasped by the robot hand, which influence the temporal consistence of PointNet feature.

4.4 Real-World Evaluation

Method	Bottle	Can	Mixed Category
EigenGrasp Oracle	0.66	0.45	N/A
EigenGrasp	0.50	0.41	N/A
Single Obj. Train	0.75 ± 0.06	0.75 ± 0.08	N/A
Multi Obj. Train	$\mathbf{0.87 \pm 0.03}$	$\mathbf{0.83 \pm 0.13}$	$\mathbf{0.73 \pm 0.12}$

Table 3: **Real-World Grasping Experiment**: This experiment consist of 3 categories, which incorporate 26 objects: 10 bottles, 6 cans, and 10 other objects in multiple mixed categories.

Settings	Original Door	Novel Door 1	Novel Door 2
Single Door Train	0.72 ± 0.07	0.60 ± 0.03	0.67 ± 0.01

Table 4: **Real-World Door Opening Experiment**: This experiment consist of 3 different doors, the first one on the left is used for training and the other two are for testing.

We perform sim2real experiments to evaluate the performance of our method in the real world. As is shown in Figure 2, we attached an Allegro hand onto a XArm-6 robot arm to grasp the object on the front table. We apply the same data-preprocessing steps for both simulated environment and real-world as mentioned in Sec.4.1.

The task execution sequence is visualized on the bottom row of Figure 1. Both Single Obj. Training and Multi Obj. Training will be evaluated in the corresponding category, while policies training with multiple objects even evaluated in the Mixed category with unseen objects. We run 10 independent trials seeds for each object-policy pair.

The real-world evaluation results are shown in Table 3 and Table 4. The policy trained in the simulator with point cloud input can directly transfer to the real-world without fine-tuning. For both two tasks, our policy can even deploy on objects that have never been seen during the training. Moreover, We find that for grasping, training on multiple objects can ensure better performance than single-object training. Compared with the EigenGrasp baseline shown in Table 3, our policies trained on both single-object and multi-object performs better, even if the EigenGrasp baseline use the object model information. Since EigenGrasp + motion planning is a open-loop manipulation policy, small error in object and scene modeling, e.g. initial object pose or object geometry, can lead to a failure in final grasp results. In contrast, our methods works in a closed-loop fashion with point cloud observation, which does not require privilege knowledge about the object.

5 Conclusion and Limitation

Limitations In our experiments, we only train and test our method on two tasks, which limits the scope of the proposed method. More dexterous manipulation tasks will be our future research direction. Another potential improvement of our method is to use Recurrent Neural Network and temporal information for policy networks. It will enable us to do long-horizon tasks.

Conclusion To the best of our knowledge, our approach is the first work to train a dexterous manipulation reinforcement learning policy with point cloud inputs that can transfer to the real world. We justified that direct sim-to-real transfer is possible for two manipulation tasks with point cloud representation.

Acknowledgments

This work was supported, in part, by grants from NSF CCF-2112665 (TILOS), NSF 1730158 CI-New: Cognitive Hardware and Software Ecosystem Community Infrastructure (CHASE-CI), NSF ACI-1541349 CC*DNI Pacific Research Platform, the Industrial Technology Innovation Program (20018112, Development of autonomous manipulation and gripping technology using imitation learning based on visualtactile sensing) funded by the Ministry of Trade, Industry and Energy of the Republic of Korea, and gifts from Meta, Google, Qualcomm. We also thanks Fanbo Xiang for suggestion on shader setting in SAPIEN, Ruihan Yang for helpful discussion on RL training.

References

[1] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[2] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3651–3657. IEEE, 2019.

[3] T. Chen, J. Xu, and P. Agrawal. A system for general in-hand object re-orientation. In *Conference on Robot Learning*, pages 297–307. PMLR, 2022.

[4] Y.-H. Wu, J. Wang, and X. Wang. Learning generalizable dexterous manipulation from human grasp affordance. *arXiv preprint arXiv:2204.02320*, 2022.

[5] 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.

[6] T. Mu, Z. Ling, F. Xiang, D. Yang, X. Li, S. Tao, Z. Huang, Z. Jia, and H. Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. *arXiv preprint arXiv:2107.14483*, 2021.

[7] X. Zhang, R. Chen, F. Xiang, Y. Qin, J. Gu, Z. Ling, M. Liu, P. Zeng, S. Han, Z. Huang, et al. Close the visual domain gap by physics-grounded active stereovision depth sensor simulation. *arXiv preprint arXiv:2201.11924*, 2022.

[8] W. Robotics. Allegro robot hand. https://www.wonikrobotics.com/research-robot-hand.

[9] J. K. Salisbury and J. J. Craig. Articulated hands: Force control and kinematic issues. *The International journal of Robotics research*, 1(1):4–17, 1982.

[10] L. Han and J. C. Trinkle. Dexterous manipulation by rolling and finger gaiting. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 730–735. IEEE, 1998.

[11] D. Rus. In-hand dexterous manipulation of piecewise-smooth 3-d objects. *The International Journal of Robotics Research*, 1999.

[12] D. Prattichizzo and J. C. Trinkle. Grasping. In *Springer handbook of robotics*, pages 955–988. Springer, 2016.

[13] A. Bicchi and V. Kumar. Robotic grasping and contact: A review. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 348–353. IEEE, 2000.

[14] J. Bohg, A. Morales, T. Asfour, and D. Kragic. Data-driven grasp synthesis—a survey. *IEEE Transactions on robotics*, 30(2):289–309, 2013.

[15] M. R. Dogar and S. S. Srinivasa. Push-grasping with dexterous hands: Mechanics and a method. 2010.

[16] O. Kroemer, S. Niekum, and G. D. Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *Journal of machine learning research*, 22(30), 2021.

[17] S. Levine, N. Wagener, and P. Abbeel. Learning contact-rich manipulation skills with guided policy search. In *IEEE International Conference on Robotics and Automation, ICRA*, pages 156–163. IEEE, 2015.

[18] A. Gupta, C. Eppner, S. Levine, and P. Abbeel. Learning dexterous manipulation for a soft robotic hand from human demonstrations. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3786–3793. IEEE, 2016.

[19] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5628–5635, 2018.

[20] I. Radosavovic, X. Wang, L. Pinto, and J. Malik. State-only imitation learning for dexterous manipulation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7865–7871. IEEE, 2020.

[21] E. Johns. Coarse-to-fine imitation learning: Robot manipulation from a single demonstration. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4613–4619. IEEE, 2021.

[22] Z. Q. Chen, K. Van Wyk, Y.-W. Chao, W. Yang, A. Mousavian, A. Gupta, and D. Fox. Learning robust real-world dexterous grasping policies via implicit shape augmentation. *arXiv preprint arXiv:2210.13638*, 2022.

[23] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *Robotics: Science and Systems*, 2018.

[24] S. Christen, S. Stevšić, and O. Hilliges. Guided deep reinforcement learning of control policies for dexterous human-robot interaction. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2161–2167. IEEE, 2019.

[25] Y. Qin, H. Su, and X. Wang. From one hand to multiple hands: Imitation learning for dexterous manipulation from single-camera teleoperation. *IEEE Robotics and Automation Letters*, 7(4): 10873–10881, 2022. doi:10.1109/LRA.2022.3196104.

[26] R. Jeong, J. T. Springenberg, J. Kay, D. Zheng, A. Galashov, N. Heess, and F. Nori. Learning dexterous manipulation from suboptimal experts. In J. Kober, F. Ramos, and C. J. Tomlin, editors, *Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 915–934. PMLR, 2020.

[27] Y. Qin, Y.-H. Wu, S. Liu, H. Jiang, R. Yang, Y. Fu, and X. Wang. Dexmv: Imitation learning for dexterous manipulation from human videos. *arXiv preprint arXiv:2108.05877*, 2021.

[28] F. Pomerleau, F. Colas, and R. Siegwart. A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends in Robotics*, 4(1):1–104, 2015.

[29] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *Robotics: Science and System*, 2017.

[30] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt. Grasp pose detection in point clouds. *The International Journal of Robotics Research*, 36(13-14):1455–1473, 2017.

[31] H. Liang, X. Ma, S. Li, M. Görner, S. Tang, B. Fang, F. Sun, and J. Zhang. Pointnetgpd: Detecting grasp configurations from point sets. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3629–3635, 2019. doi:10.1109/ICRA.2019.8794435.

[32] A. Mousavian, C. Eppner, and D. Fox. 6-dof graspnet: Variational grasp generation for object manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2901–2910, 2019.

[33] Y. Qin, R. Chen, H. Zhu, M. Song, J. Xu, and H. Su. S4g: Amodal single-view single-shot se(3) grasp detection in cluttered scenes. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 53–65. PMLR, 2020.

[34] P. Ni, W. Zhang, X. Zhu, and Q. Cao. Pointnet++ grasping: Learning an end-to-end spatial grasp generation algorithm from sparse point clouds. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3619–3625, 2020. doi:10.1109/ICRA40945.2020.9196740.

[35] C. Wu, J. Chen, Q. Cao, J. Zhang, Y. Tai, L. Sun, and K. Jia. Grasp proposal networks: An end-to-end solution for visual learning of robotic grasps. *Advances in Neural Information Processing Systems*, 33:13174–13184, 2020.

[36] W. Wei, Y. Luo, F. Li, G. Xu, J. Zhong, W. Li, and P. Wang. Gpr: Grasp pose refinement network for cluttered scenes. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4295–4302. IEEE, 2021.

[37] W. Wei, D. Li, P. Wang, Y. Li, W. Li, Y. Luo, and J. Zhong. Dvgg: Deep variational grasp generation for dextrous manipulation. *IEEE Robotics and Automation Letters*, 2022.

[38] L. Wang, Y. Xiang, W. Yang, A. Mousavian, and D. Fox. Goal-auxiliary actor-critic for 6d robotic grasping with point clouds. In *Conference on Robot Learning*, pages 70–80. PMLR, 2022.

[39] S. Andrews and P. G. Kry. Goal directed multi-finger manipulation: Control policies and analysis. *Computers & Graphics*, 2013.

[40] J. Varley, J. Weisz, J. Weiss, and P. Allen. Generating multi-fingered robotic grasps via deep learning. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 4415–4420. IEEE, 2015.

[41] S. Brahmbhatt, A. Handa, J. Hays, and D. Fox. Contactgrasp: Functional multi-finger grasp synthesis from contact. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2386–2393. IEEE, 2019.

[42] Q. Lu, K. Chenna, B. Sundaralingam, and T. Hermans. Planning multi-fingered grasps as probabilistic inference in a learned deep network. In *Robotics Research*, pages 455–472. Springer, 2020.

[43] Y. Chebotar, O. Kroemer, and J. Peters. Learning robot tactile sensing for object manipulation. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3368–3375, 2014. doi:10.1109/IROS.2014.6943031.

[44] W. Yuan, S. Dong, and E. H. Adelson. Gelsight: High-resolution robot tactile sensors for estimating geometry and force. *Sensors*, 17(12):2762, 2017.

[45] R. Calandra, A. Owens, D. Jayaraman, J. Lin, W. Yuan, J. Malik, E. H. Adelson, and S. Levine. More than a feeling: Learning to grasp and regrasp using vision and touch. *IEEE Robotics and Automation Letters*, 3(4):3300–3307, 2018.

[46] A. Murali, Y. Li, D. Gandhi, and A. Gupta. Learning to grasp without seeing. In *International Symposium on Experimental Robotics*, pages 375–386. Springer, 2018.

[47] M. Lambeta, P.-W. Chou, S. Tian, B. Yang, B. Maloon, V. R. Most, D. Stroud, R. Santos, A. Byagowi, G. Kammerer, et al. Digit: A novel design for a low-cost compact high-resolution tactile sensor with application to in-hand manipulation. *IEEE Robotics and Automation Letters*, 5(3):3838–3845, 2020.

[48] M. A. Lee, Y. Zhu, P. Zachares, M. Tan, K. Srinivasan, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg. Making sense of vision and touch: Learning multimodal representations for contact-rich tasks. *IEEE Transactions on Robotics*, 36(3):582–596, 2020.

[49] R. Bhirangi, T. Hellebrekers, C. Majidi, and A. Gupta. Reskin: versatile, replaceable, lasting tactile skins. *arXiv preprint arXiv:2111.00071*, 2021.

[50] H. Van Hoof, T. Hermans, G. Neumann, and J. Peters. Learning robot in-hand manipulation with tactile features. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 121–127. IEEE, 2015.

[51] N. Vulin, S. Christen, S. Stevšić, and O. Hilliges. Improved learning of robot manipulation tasks via tactile intrinsic motivation. *IEEE Robotics and Automation Letters*, 6(2):2194–2201, 2021.

[52] S. H. Huang, M. Zambelli, J. Kay, M. F. Martins, Y. Tassa, P. M. Pilarski, and R. Hadsell. Learning gentle object manipulation with curiosity-driven deep reinforcement learning. *arXiv preprint arXiv:1903.08542*, 2019.

[53] H. Xu, Y. Luo, S. Wang, T. Darrell, and R. Calandra. Towards learning to play piano with dexterous hands and touch. *arXiv preprint arXiv:2106.02040*, 2021.

[54] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, L. Yi, A. X. Chang, L. J. Guibas, and H. Su. SAPIEN: A simulated part-based interactive environment. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[55] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[56] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pages 510–517. IEEE, 2015.

[57] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[58] M. Ciocarlie, C. Goldfeder, and P. Allen. Dexterous grasping via eigengrasps: A low-dimensional approach to a high-complexity problem. In *Robotics: Science and systems manipulation workshop-sensing and adapting to the real world*, 2007.

[59] A. T. Miller and P. K. Allen. Graspit! a versatile simulator for robotic grasping. *IEEE Robotics & Automation Magazine*, 11(4):110–122, 2004.

[60] J. J. Kuffner and S. M. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 995–1001. IEEE, 2000.

[61] I. A. Sucan, M. Moll, and L. E. Kavraki. The open motion planning library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, 2012.

[62] J. R. Napier. The prehensile movements of the human hand. *The Journal of bone and joint surgery. British volume*, 38(4):902–913, 1956.

[63] K. Mamou and F. Ghorbel. A simple and efficient approach for 3d mesh approximate convex decomposition. In *2009 16th IEEE international conference on image processing (ICIP)*, pages 3501–3504. IEEE, 2009.

Appendices

In this supplementary material, we provide more details of the implementation. For more visualization, please see our video and project page: https://yzqin.github.io/dexpoint.

A Environment Details

Object Set. For single-object experiments, we use the single object mesh from the YCB dataset for training and testing. For multi-object experiments, we chooses 10 objects from the ShapeNet dataset [55] for training, and another 40 objects for testing in simulation. The test objects of real-world experiments are described in the main paper.

Object Pre-processing. To use the above object models in the simulation experiments, we apply the object preprocessing. The preprocessing includes two steps: scaling and convex decomposition. The scaling step is to ensure that the object size is within a proper ranger for manipulation. The YCB objects [56] are captured by the real scan, so the scale of object mesh from YCB dataset aligns with the real counterpart and can be used for robot manipulation directly. Different from the YCB dataset, the ShapeNet dataset does not contain any scale information, e.g. the mug in ShapeNet can be larger than the robot. Therefore, we scale each object based on the diagonal length of its bounding box. For each category, we manually select a diagonal length so that all object instances from the category will have the same bounding-box diagonal length after scaling. Besides, we will not use objects with non-manifold geometry to avoid instability in physical simulation. The next step is convex decomposition. We use V-HACD [63] to decompose both YCB object and ShapeNet object into convex parts with default parameters. We will not use the object with more than 40 parts after convex decomposition for both stability and efficiency of the physical simulation.

Reward. As is mentioned in Section 3.2 of the main paper, the overall reward for our task is composed of four parts: reach, contact, lift, and action penalty.

$$\mathcal{R} = w_{\text{reach}} r_{\text{reach}} + w_{\text{contact}} r_{\text{contact}} + w_{\text{lift}} r_{\text{lift}} + w_{\text{penalty}} r_{\text{penalty}}. \tag{5}$$

In the implementation, we set the lift reward as the the difference between object current height and object initial height $r_{\text{lift}} = h_{\text{current}} - h_{\text{init}}$. The action penalty reward $r_{\text{penalty}} = -||a||_2^2$. For reaching reward, it consists of the distance between object and target, the distance between finger tip and the target. The weight for the four reward terms are: $w_{\text{reach}} = 1$, $w_{\text{contact}} = 0.5$, $w_{\text{lift}} = 10$, $w_{\text{penalty}} = 0.01$. Recall that the lift reward r_{lift} is set to 0 if the contact reward r_{contact} is 0. Intuitively, it ensures that the robot can only receive the reward for lifting the object up if the robot hand is in good contact with the object. This design prevents the agent from using a large force to knock the object into the air or using unstable contact to lift the object up.

B Learning Details

RL Training. We use on-policy RL training for setting except the distilling experiments. We select PPO as the RL algorithm to train the point cloud based manipulation policy. The hyper-parameters of the PPO are shown in Table 5.

Network Architecture The RL agent use the PointNet based architecture as the visual backbone. As shown in Figure 3 of the main paper, we first concatenate the visual features from PointNet and the propriception feature from the MLP. This concatenated feature is then shared by both value network and policy network to predict value and action. The details for the network architecture are shown in Table 6.

Parameter	Value
Mini-Batch Size	500
Learning Rate	3e-4
Clip Range	0.8
Horizon	200
Epoch	10
Steps per Iteration	10

Table 5: PPO Parameters.

Module	Architecture	Output Dim
Visual Feature Extractor	PointNet Local Channel: (64, 128, 256)	256
	PointNet Global Channel: (256,)	256
State Feature Extractor	MLP: (64, 64)	64
Actor	MLP: (64, 64)	64
Critic	MLP: (64, 64)	64

Table 6: Network Architecture.