

RiEMann: Near Real-Time SE(3)-Equivariant Robot Manipulation without Point Cloud Segmentation

Chongkai Gao¹, Zhengrong Xue², Shuying Deng³, Tianhai Liang⁴, Siqu Yang⁵, Lin Shao¹, Huazhe Xu^{2,6,7}

¹National University of Singapore ²Tsinghua IIIS ³Department of Automation, Tsinghua University

⁴Harbin Institute of Technology, Shenzhen ⁵Department of Electrical Engineering, Tsinghua University

⁶Shanghai AI Lab ⁷Shanghai Qizhi Institute

Abstract—We present RiEMann, an end-to-end near Real-time SE(3)-Equivariant Robot Manipulation imitation learning framework from scene point cloud input. Compared to previous methods that rely on descriptor field matching, RiEMann directly predicts the target poses of objects for manipulation without any object segmentation. RiEMann learns a manipulation task from scratch with 5 to 10 demonstrations, generalizes to unseen SE(3) transformations and instances of target objects, resists visual interference of distracting objects, and follows the near real-time pose change of the target object. The scalable action space of RiEMann facilitates the addition of custom equivariant actions such as the direction of turning the faucet, which makes articulated object manipulation possible for RiEMann. In simulation and real-world 6-DOF robot manipulation experiments, we test RiEMann on 5 categories of manipulation tasks with a total of 25 variants and show that RiEMann outperforms baselines in both task success rates and SE(3) geodesic distance errors on predicted poses (reduced by 68.6%), and achieves a 5.4 frames per second (FPS) network inference speed. Code and video results are available on <https://riemann-web.github.io/>.

I. INTRODUCTION

Learning from demonstrations is an effective and convenient mechanism for visual robot manipulation tasks [2, 41]. However, most current algorithms for learning from demonstrations suffer from low data efficiency and generalization ability to new situations. For example, current visual imitation learning algorithms require around 100 demonstrations [37, 36, 6] to learn a simple manipulation task such as picking up a mug and placing it on a rack, and cannot generalize to new object poses beyond the training distribution. Although previous works have tried to tackle these problems by data augmentation [33, 31] or contrastive learning on demonstrations [45, 32, 17], they rely on task-specific domain knowledge and have no algorithmic guarantees to generalize to unseen object poses.

Symmetries pervade the physical real world, and the same is true in robot manipulation problems. Exploiting these symmetries can improve the sample efficiency and generalization ability of robot learning algorithms [3]. Roto-translation equivariance is one of the most common types of symmetry for robot manipulation tasks. It denotes that robot actions can transform with the same SE(3) transformations as the target objects. Although some works [56, 51, 30] have successfully integrated SE(2)-equivariance on visual desktop top-down-view pick-and-place tasks, the adoption of SE(3)-equivariance in manipulation tasks still remains constrained by various limitations. Effectively harnessing the symmetry inherent in three-

dimensional space has the potential to significantly broaden the scope of robotic learning algorithms.

As pioneering works, NDFs [47, 48] propose to first pretrain a category-level object descriptor field with SE(3)-equivariant backbones [11] and then output robot actions with energy-based optimization to match different descriptor fields. EquivAct [55] inherits this two-stage idea but learns a neural network to output actions. However, these works require well-segmented object point clouds from scene inputs, which is nontrivial in real-world robot manipulation tasks, and the two-stage paradigm makes it hard to fine-tune on new object categories. On the other hand, EDFs [42, 43] improve the above idea to learn a descriptor field with *local* SE(3)-equivariant backbones [18, 35], which avoids fine-grained object segmentation from the scene and enables the end-to-end training procedure. However, they still need to separate the robot end-effector and the grasped target object point cloud from the scene to perform the time-consuming (around 10s) descriptor field matching process for every inference, which makes EDFs not segmentation-free and prevents them from being applied to real-time manipulation tasks. Moreover, the performance of EDFs highly relies on the optimization results from the whole scene energy field, which is often unstable in SE(3) generalization cases in real-world manipulation tasks.

The key problem comes from the descriptor field matching paradigm. It leads to the time-consuming energy-based optimization on the whole SE(3) space, which significantly increases the training difficulty of manipulation tasks that actually only require a target pose prediction. Inspired by previous works on SE(2)-equivariant robot learning [51], using a network that directly predicts actions rather than performing optimization could solve this issue. However, extending this idea to our SE(3) group faces two problems: 1) SE(3)-equivariant models themselves consume considerable computational complexity and memory usage [28, 15, 21], not to mention the daunting burdens from end-to-end processing of scene-level point cloud; 2) Designing an SE(3)-equivariant action space for end-to-end learning is non-trivial. Although SE(3)-equivariant backbones [11, 18, 35] can learn equivariant features, these features are restricted to specific formats that are usually incompatible with common 3D rotation representations. For example, steerable group representations outputted from SE(3)-transformers [18] are incompatible with quaternion, a common rotation action representation. This is why

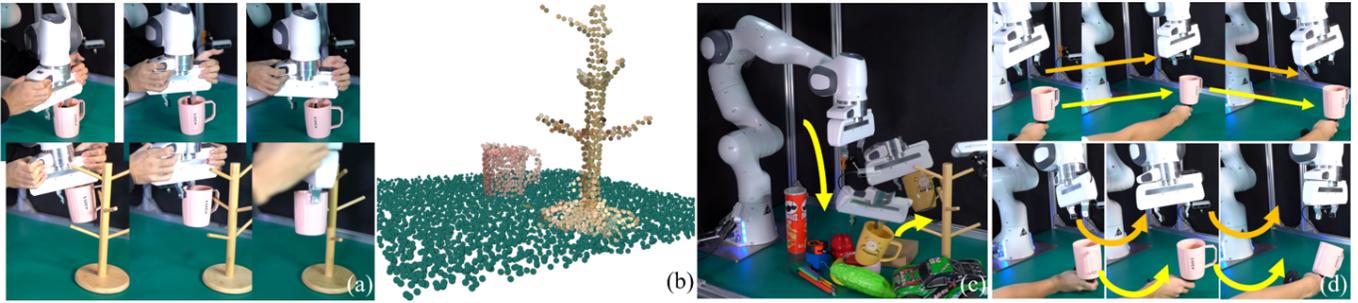


Fig. 1: Overview of RiEMann. (a) Given 5 to 10 demonstrations of restricted object poses (The mug remains standing and only rotates in 90 degrees around the z-axis) of the task *Mug on Rack* and (b) with the full scene point cloud as input without any segmentation, (c) RiEMann can generalize to any local SE(3)-equivariant transformations of target objects, to new instances of target objects, be robust to distracting objects in the scene, and (d) has the near real-time following ability of target objects.

previous works bypass this action space parameterization problem and instead match different feature fields [47, 42, 48, 43] to define actions.

In this work, we present RiEMann, the first near Real-time SE(3)-Equivariant robot Manipulation framework from point cloud inputs without any segmentation. RiEMann leverages local SE(3)-equivariant backbones [18] and learns a policy that directly outputs SE(3)-equivariant actions. RiEMann tackles the computational complexity problem by firstly learning an SE(3)-invariant saliency map on the input scene point cloud to extract a small region of interest, and then training the main SE(3)-equivariant policy on the extracted point cloud. This mechanism greatly decreases the computational complexity since the main policy occupies the main computing resources and memory usage. Next, RiEMann tackles the action parameterization problem by using an SE(3)-invariant vector field as the target point affordance map for translational actions and using three SE(3)-equivariant vector fields as three bases of the target rotation matrix for the rotational action under steerable group representation theory [49, 3, 18], and combining these bases to acquire a legal rotation matrix with Gram-Schmidt orthogonalization. We also prove that other common parameterizations for SO(3), such as axis-angle, quaternion, and Euler angle, are not trainable SE(3)-equivariant representations.

In both simulation and real-world experiments, we demonstrate that RiEMann can solve various manipulation tasks with only 5 to 10 demonstrations for each task, generalize to unseen poses and instances of target objects, resist visual interference of distracting objects, and have the near real-time following capacity, as illustrated in Figure 1. We also show that our designed action space can be easily applied to articulated object manipulation tasks by adding another type-1 action on policy networks. On 5 manipulation tasks with a total of 25 different task settings, RiEMann outperforms baselines on both success rates as well as the SE(3) geodesic distance errors (reduced the geodesic error by **68.6%**) and a **5.4 FPS** inference speed, demonstrating the advantages of the end-to-end supervised learning paradigm of RiEMann over previous field-matching methods.

II. RELATED WORKS

A. Group Equivariant Neural Networks

Exploiting groups of symmetry pervading in data and incorporating them into deep neural networks has been the focus of many studies since it can improve generalization and data efficiency. Equivariant neural networks, which are first introduced into CNNs [8], extract symmetries from various kinds of data. According to [3, 22], most equivariant models can be divided into two categories: regular group representation networks and steerable group representation networks. The former seeks to define equivariant convolution filters [8, 14], attention mechanism [28], or message passing mechanism [19] as functions on groups, while the latter uses irreducible group representations [12] with spherical harmonics as an equivariant basis to perform message passing [9, 49, 18, 35]. Some other works [11] design special non-linear kernels to achieve equivariance on the SO(3) group. The theory of equivariant networks on homogeneous spaces is formalized in [10] with vector bundles and the group representation theory, and the implementation of constructing equivariant CNN layers for arbitrary matrix groups is given in [16]. Equivariant networks have been applied for different groups including SO(2) [9, 13, 39], SO(3) [49, 52, 1, 11], SO⁺(1,3) [4], and SE(3) [26, 42].

B. Equivariant Robot Manipulation

Researchers have been exploring equivariant models for robotic manipulation tasks to improve generalizability and sample efficiency. Pioneering works [56, 44, 25, 51, 50, 57, 27] learn equivariant representations of objects or the scene with equivariant networks to get SO(2)- or SE(2)-equivariance for desktop manipulation tasks with imitation learning or reinforcement learning. NDFs [47, 48, 55] leverages Vector Neurons [11] to acquire SE(3)-equivariant category-level object representations from point cloud of objects for downstream imitation learning, while EDFs [42, 43] uses SE(3)-Transformer [18] and Equiformers [35] to directly learn SE(3)-equivariant representations from point clouds of the scene. EFEM [34] designs an EM-like SE(3)-equivariant object segmentation refinement procedure from the scene, but it can only be trained on well-segmented data. Some works use soft

loss functions with geometric constraints [7, 30] rather than equivariant models to acquire equivariant representations, but they cannot guarantee equivariance. In this work, we leverage SE(3)-transformers [18] and design an end-to-end learning paradigm to predict target actions from scene point cloud input in near real time.

III. BACKGROUND AND PROBLEM FORMULATION

A. Problem Formulation

Let a colored point cloud with N points be $\mathbf{P} = \{(x_1, c_1), \dots, (x_N, c_N)\} \in \mathbb{R}^{N \times 6}$, where $x_i \in \mathbb{R}^3$ is the position and $c_i \in \mathbb{R}^3$ is the RGB color of the i -th point. For a manipulation task \mathcal{T} , a policy f_θ parameterized by θ is trained to predict the target pose of robot end-effector $\mathbf{T} = \{\mathbf{R}, \mathbf{t}\} \in SE(3)$ with the scene point cloud \mathbf{P} , where $\mathbf{R} \in SO(3)$ is the 3D rotation action and $\mathbf{t} \in \mathbb{R}^3$ is the 3D translation action. We assume a motion planner with collision avoidance is used to execute the action \mathbf{T} , as in [46, 47, 42].

For each task \mathcal{T} , a set of demonstrations $\mathcal{D} = \{(\mathbf{P}_i, \mathbf{T}_i)\}_{i=1}^m$ that consists of m pairs of data is provided to train the policy to output $\hat{\mathbf{T}} = f_\theta(\mathbf{P})$, where the hat symbol indicates the predicted output. We aim to train the policy f_θ only with 5 to 10 demonstrations and then deploy the learned policy in a setup with unseen target object poses, distracting objects, and new object instances, which relies on the following local SE(3)-equivariance property of our policy.

B. Local SE(3)-Equivariance

Let $g \in SE(3)$ be an element of the Special Euclidean Group in three dimensions (SE(3)) and $T_g : \mathcal{X} \rightarrow \mathcal{X}$ be a transformation of g on a space \mathcal{X} . A function $f : \mathcal{X} \rightarrow \mathcal{Y}$ is called SE(3)-equivariant if there exists a transformation $S_g : \mathcal{Y} \rightarrow \mathcal{Y}$ such that [18]:

$$S_g[f(x)] = f[T_g(x)], \quad \forall g \in SE(3), x \in \mathcal{X}. \quad (1)$$

It is worth noting that if S_g is the identity mapping: $S_g(f(x)) = f(x)$, f becomes SE(3)-invariant. In our robot manipulation tasks, f is the trained policy, \mathcal{X} is the space of the scene point cloud \mathbf{P} , \mathcal{Y} is the space of target pose \mathbf{T} , T_g is the space of any SE(3) transformation $T_g = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$ on \mathbf{P} , and S_g is the SE(3) transformation on the target pose \mathbf{T} . The SE(3)-equivariance is achieved by using SE(3)-equivariant networks to parameterize f . Note although S_g can be inconsistent with T_g (they can be on different orbits [10, 3]), researchers in the robotics community usually seek the same transformation [10, 3, 26] on the input and output, i.e., we here define $S_g = T_g$. We leave the study of more flexible S_g to future works.

Local SE(3)-equivariance [42, 43] refers that T_g can be applied on only part(s) of the input scene point cloud \mathbf{P} rather than the whole scene input, as illustrated in Figure 1. This is more practical for robot manipulation tasks, in which we want the policy to be only equivariant to the target objects and robust to the scene and other distracting objects. Local equivariance can be achieved by equivariant networks with

local mechanisms such as convolutional kernels [56, 57] or local message-passing mechanisms [18, 49]. In this work, we choose to use SE(3)-transformers [18] that belong to the latter category, as introduced below.

C. Group Representations and SE(3)-Transformers

T_g and S_g above are called *group representations*. Formally, a group representation \mathbf{D} of a group G is a map from group G to the set of $N \times N$ invertible matrices $GL(N)$, and it satisfies $\mathbf{D}(g)\mathbf{D}(h) = \mathbf{D}(gh), \forall g, h \in G$. Specifically, any representation of SO(3) group $\mathbf{D}(\mathbf{R})$ for $\forall \mathbf{R} \in SO(3)$ can be block-diagonalized as the direct sum of orthogonal Wigner-D matrices $\mathbf{D}_l(\mathbf{R}) \in \mathbb{R}^{(2l+1) \times (2l+1)}$ [18]:

$$\mathbf{D}(\mathbf{R}) = \mathbf{Q}^T \left[\bigoplus_l \mathbf{D}_l(\mathbf{R}) \right] \mathbf{Q}, \quad l \in \{0, 1, 2, \dots\}, \quad (2)$$

where \mathbf{Q} is the orthogonal $N \times N$ change-of-basis matrix. Vectors transforming according to $\mathbf{D}_l(\mathbf{R})$ are called *type- l vectors* [3]. Type-0 vectors are invariant under rotations ($\mathbf{D}(\mathbf{R}) = \mathbf{I}$) and type-1 vectors (3d space vectors) rotate according to 3D rotation matrices. Note, type- l vectors have length $2l + 1$.

Given a point cloud \mathbf{P} , we can assign a type- l vector to each point $x \in \mathbf{P}$ to get a type- l vector field $\mathbf{f}_l : \mathbb{R}^3 \rightarrow \mathbb{R}^{2l+1}$. Same or different type- l vector fields can be stacked to get a complex vector field. SE(3)-transformers [18] are neural networks that map point cloud vector fields to point cloud vector fields with the same point number, and they are designed to achieve *local SE(3)-equivariance* for any learned type- l field:

$$\mathbf{D}_l(\mathbf{R})\mathbf{f}_l(x) = \mathbf{f}_l(Tx), \quad \forall x \in \mathbf{P}, \forall T = (\mathbf{R}, \mathbf{t}) \in SE(3). \quad (3)$$

We can see that the SO(3)-equivariance is achieved by the Wigner-D matrices $\mathbf{D}_l(\mathbf{R})$, while the translational T(3)-invariance is achieved by the local mechanisms of SE(3)-transformers, i.e., the message-passing mechanism.

In short, SE(3)-transformers can transform the input vector fields to the output vector fields with local SE(3)-equivariance. To leverage this property for robot manipulation tasks, we need to design special input and output vector fields to satisfy task requirements, as discussed in the following section.

IV. SE(3)-EQUIVARIANT ROBOT MANIPULATION

A. The Main Idea of RiEMann

According to the above discussion, the main question lies in designing the input and output vector fields for SE(3)-transformers. As comparisons, NDFs [47, 48] output several type-0 vector fields, while EDFs [42, 43] learn one type-0 vector field and a stack of high type- l vector fields. However, they do not use learned vector fields as actions. They use field-matching optimization between two vector fields (end-effector and target object) to get the desired robot action.

In fact, this process is not only time-consuming, but also increases the difficulty of training, since the field-matching process is doing optimization given the energy-based model, which can be seen as a *generative modeling* process that

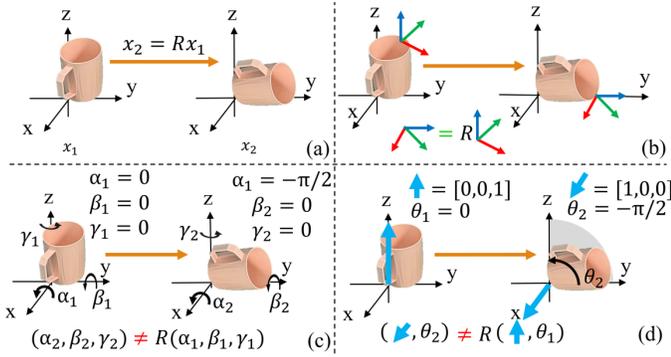


Fig. 2: Illustration of different 3D rotation representations under type- l parameterizations. (a) The initial point cloud x_1 is transformed to x_2 with 3D rotation R ; (b) Using three type-1 vectors to represent a Rotation Matrix. It transforms with the same transformation R as the input; (c) (d) Using three type-0 vectors to represent the Euler Angles, and using one type-1 vector and one type-0 vector to represent Axis-angle. They cannot be transformed with the same transformation R as the input, so they are not SE(3)-equivariant parameterizations.

actually learns a target pose probability distribution $p(T)$, $T \in SE(3)$. Instead, we aim to design special input and output vector fields to make the outputted vector fields from networks directly the desired action, which is a much simpler process and can lead to higher precision. In this way, our model becomes a *discriminative* model that can be trained with supervised objectives.

B. Design Choices of Vector Fields

There are two main requirements for the design of input and output vector fields: a) all the input (scene point cloud) and output (robot actions) should be contained in the vector fields; b) the predicted actions parameterized by output vector fields are SE(3)-equivariant to the input transformation in theory. For the input design, each point in our point cloud \mathbf{P} contains the positions and RGB information. Since SE(3)-transformer is actually T(3)-invariant, thus we should only use the T(3)-invariant features, or visual features, as the input, to ensure T(3)-invariance. Thus here we only use color information for input. Note if we want to make our model generalize to new instances with different colors, we need to do color randomization. RGB can be seen as 3 type-0 vectors, so our input is a direct sum of three vector fields:

$$\mathbf{f}_{in}(x) = \bigoplus_{i=1}^3 \mathbf{f}_0^i(x), \forall x \in \mathbf{P}. \quad (4)$$

For the output, our model predicts the end-effector target pose $\mathbf{T} = \{\mathbf{R}, \mathbf{t}\} \in SE(3)$, which can be decomposed into target position \mathbf{t} and target rotation \mathbf{R} . For target position \mathbf{t} , we cannot directly output the 3D target position vector since SE(3)-transformers are only able to predict T(3)-invariant features. Thus we choose to learn one type-0 vector field as an affordance map [29, 38] on point clouds, and use softmax

on the affordance map to perform weighted sum on all point positions to get the translational action \mathbf{t} . For target orientation \mathbf{R} , people usually use Euler angle, quaternion, rotation matrix, or axis-angle as robot action parameterization. Although all these pose parameterizations can satisfy the first requirement (representing output action information), we need to ensure that after parameterizing them with different type- l vectors, they are SE(3)-equivariant to the input transformation. We answer this question with the following theorems. All proofs are in Appendix B.

Theorem 1 *Rotation matrices, represented by three type-1 vectors, are SE(3)-equivariant parameterization of rotation actions.*

Theorem 2 *There is no SE(3)-equivariant vector field representation for Euler angle, quaternion, and axis-angle.*

The intuitions behind these theorems are illustrated in Figure 2. Rotation matrices, represented by three type-1 vectors, are the only SE(3)-equivariant rotation representation to the input rotation transformation. Note although a rotation matrix can be transformed to any other rotational representations (e.g., using inverse Rodrigues' formula to get an axis-angle representation), this process will be outside the network and is not trainable, thus cannot be parameterized by type- l vectors.

In summary, RiEMann learns one type-0 vector field (target point heatmap) for target position \mathbf{t} and three type-1 vector fields for target rotation \mathbf{R} , which is a direct sum of 4 vector fields:

$$\mathbf{f}_{out}(x) = \mathbf{f}_0(x) + \bigoplus_{j=1}^3 \mathbf{f}_1^j(x), \quad \forall x \in \mathbf{P}. \quad (5)$$

As a comparison, EquivAct [55] uses one type-1 vector field as output (the end-effector velocity), which means EquivAct only supports 3-DOF manipulation tasks. Instead, RiEMann can support 6-DOF manipulation through the action space defined by Equation 5.

Equation 5 also shows that if one wants to predict other physical properties such as 3D moving directions (type-1), forces (type-1), gripper aperture (type-0), or more than one target object pose, it is convenient to directly expand the output vector fields with corresponding type- l vector fields to add desired properties. In experiments, we demonstrate how to add moving directions for an articulated object manipulation task (tuning faucet, Figure 5b). Thus RiEMann is a highly scalable framework for general SE(3)-equivariance manipulation.

The next question is how to use network to implement equation 5 and transform the output vector fields that contain features on each point to a single output action while keeping SE(3)-equivariance.

C. Network Design

As we point out in the introduction, the first problem of leveraging equivariant networks for scene point cloud input is to reduce the heavy computational cost. A scene point cloud for robot manipulation tasks usually contains thousands to tens of thousands of points (in our case, 8192 points), and end-to-end learning on such a large number of points brings

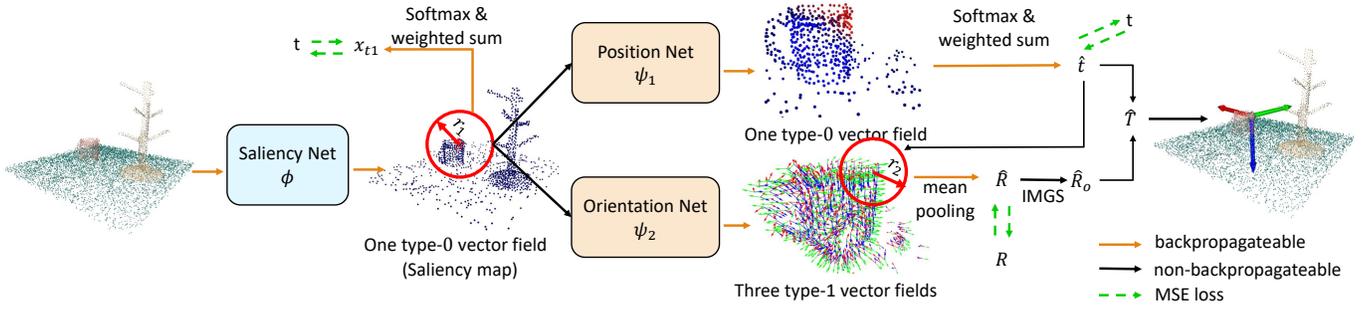


Fig. 3: Pipeline of RiEMann. For a point cloud input of a scene, a type-0 saliency map is firstly outputted by an $SE(3)$ -invariant backbone ϕ to get a small point cloud region \mathbf{B}_{ROI} , and an $SE(3)$ -equivariant policy network that contains a translational heatmap network ψ_1 and an orientation network ψ_2 predicts the action vector fields on the points of \mathbf{B}_{ROI} . Finally, we perform softmax, mean pooling, and Iterative Modified Gram-Schmidt orthogonalization to get the target action \mathbf{T} .

Algorithm 1 RiEMann Training

Input: Demonstrations $\{(\mathbf{P}_i, \mathbf{T}_i)\}_{i=1}^M$, initialized models ϕ , ψ_1 , ψ_2 , hyperparameters r_1 and r_2 , epochs n .

- 1: **for** $iter = 0$ to $n - 1$ **do**
- 2: Sample a batch of m demonstrations $\{(\mathbf{P}_i, \mathbf{T}_i)\}_{i=1}^m$, where $\mathbf{T}_i = (\mathbf{R}_i, \mathbf{t}_i)$
- 3: Predict the saliency map $\mathbf{f}_s(x) = \phi(x), x \in \mathbf{P}_i$
- 4: Get x_{t1} by doing weighted sum on \mathbf{P} with the softmax weight from $\mathbf{f}_s(x)$
- 5: Get \mathbf{B}_{ROI} centered on x_{t1} with radius r_1
- 6: Predict $\mathbf{f}_t(x) = \psi_1(x), \mathbf{f}_R(x) = \psi_2(x), \forall x \in \mathbf{B}_{ROI}$
- 7: Get $\hat{\mathbf{t}}$ as the weighted position of $\mathbf{f}_t(x)$ and get $\hat{\mathbf{R}}$ by mean pooling on $\mathbf{f}_R(x)$ on points centered at $\hat{\mathbf{t}}$ with the radius r_2
- 8: Normalize each type-1 vector of $\hat{\mathbf{R}}$
- 9: Update ϕ , ψ_1 , and ψ_2 with $\mathcal{L} = \sum_{i=0}^m [\sum_{j=1}^N (\mathbf{t}_i - \hat{\mathbf{t}}_i)^2 + \sum_{k=1}^{N_B} ((\mathbf{t}_i - \hat{\mathbf{t}}_i)^2 + (\mathbf{R}_i - \hat{\mathbf{R}}_i)^2)]$
- 10: **end for**

Output: Trained models ϕ , ψ_1 , and ψ_2

$\psi_1(x)$ and an orientation network $\psi_2(x)$ for $\forall x \in \mathbf{B}_{ROI}$ as policy networks for action prediction. For the translational part, ψ_1 outputs one type-0 affordance $\mathbf{f}_t(x) = \mathbf{f}_0(x), x \in \mathbf{B}_{ROI}$. We then perform softmax on $\mathbf{f}_t(x)$ as weight and multiply them to the positions of all points of \mathbf{B}_{ROI} to get the output translational action $\hat{\mathbf{t}}$. For the orientation part, ψ_2 outputs three type-1 vector fields $\mathbf{f}_R = \bigoplus_{i=1}^3 \mathbf{f}_1^i(x), x \in \mathbf{B}_{ROI}$. These three vectors can be seen as the predicted three axes of a rotation matrix, despite they are not orthogonal to each other yet. To get the target rotation matrix, although we can directly perform mean pooling on all vector fields for all points in \mathbf{B}_{ROI} , we find this is often unstable since there are still many points that do not belong to the target object in \mathbf{B}_{ROI} . Instead, we here select the points from a smaller region that are centered on $\hat{\mathbf{t}}$ with a radius r_2 , and perform mean pooling on these points to get the output orientation action $\hat{\mathbf{R}}$, as in the right side of Figure 3. During training, we use the translation action \mathbf{t} in the demonstrations as supervision for both $\phi(x)$ and $\psi_1(x)$, and three orientation axes of \mathbf{R} to train $\psi_2(x)$. The training algorithm of RiEMann is shown in Algorithm 1.

huge burdens for GPU memory and a long training time. Meanwhile, the training difficulty also increases since most of the points are not informative. For example, EDFs [42] only support batch size = 1 for training on an Nvidia RTX3090 GPU. The key is to eliminate the large amount of redundant information contained in the scene point cloud, since the target objects only occupy a small part of the input. Inspired by [57, 56], we can first learn a $SE(3)$ -invariant network to get a relatively small region of point cloud from the scene point cloud for downstream policy networks. Note this does not mean a fine-grained object segmentation. In this work, we employ an $SE(3)$ -transformers $\phi(x)$ to output one type-0 vector field $\mathbf{f}_s(x) = \mathbf{f}_0(x), x \in \mathbf{P}$ for each point as a *saliency map*, and extract a point cloud ball \mathbf{B}_{ROI} with a predefined radius r_1 centered on the point with the highest value, as illustrated in the left part of Figure 3.

With \mathbf{B}_{ROI} as input, we then employ another two $SE(3)$ -transformer modules to learn a translational action network

For the orientation output $\hat{\mathbf{R}}$, although we get three type-1 vectors from ψ_2 , they are not necessarily orthogonal to each other, in which case they cannot form a legal rotation matrix for robot control. In this work, we perform Iterative Modified Gram-Schmidt orthogonalization (IMGS) [24] on the orientation output to get an orthogonal matrix $\hat{\mathbf{R}}_o$ for actual robot control, as summarized in Appendix A.

For tasks with multiple stages such as picking up the mug and placing it on the rack, the execution quality of the previous stage will affect the execution effect of the later stage, thus directly predicting the target object pose is not optimal for the second stage of these tasks. In this work, we predict both the mug pose and the rack pose *after* picking up the mug, and calculate the pose transformation $\hat{\mathbf{T}} = \mathbf{T}_{rack} \mathbf{T}_{mug}^{-1}$ as the target action for placing, as illustrated in Figure 5a. Note this is a direct calculation and is much faster than descriptor field matching in [47, 42, 43].

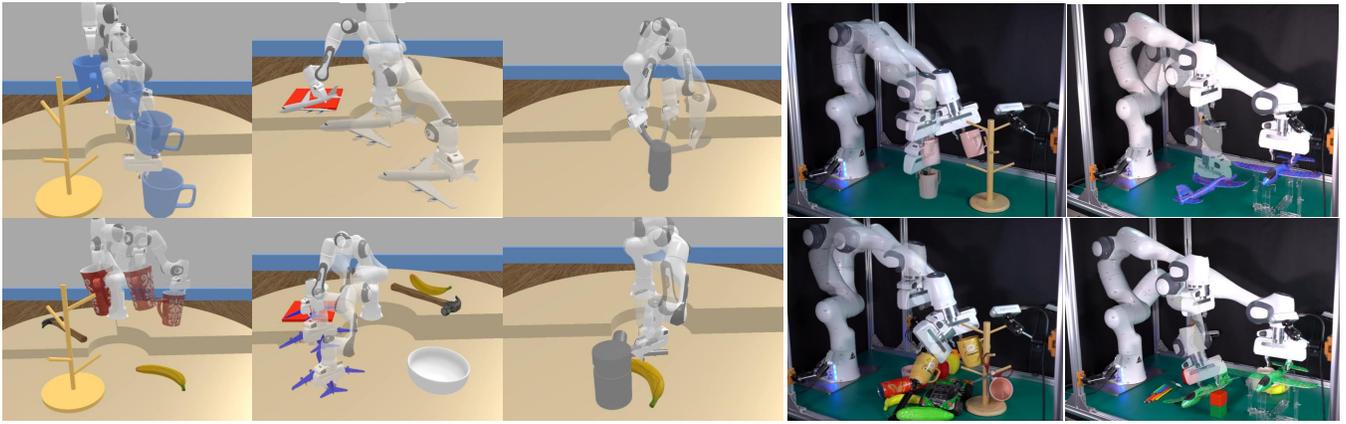


Fig. 4: Simulation and real-world environments. Top: the training environment settings of simulation tasks *Mug on Rack*, *Plane on Shelf*, *Turning Faucet*, and real world tasks *Mug on Rack*, and *Plane on Shelf*. Bottom: the *ALL* testing case of tasks, where the target object is a new instance and in a new pose, and distracting objects are added in the environments.

D. Implementation Details

We use NVIDIA’s implementation [40] of SE(3)-transformers that is more efficient than the original official implementation [18]. We also perform voxel downsampling as well as random color dropping and jittering for color augmentation. Detailed input and output processing and network hyperparameters are listed in Appendix C.

V. EXPERIMENTS

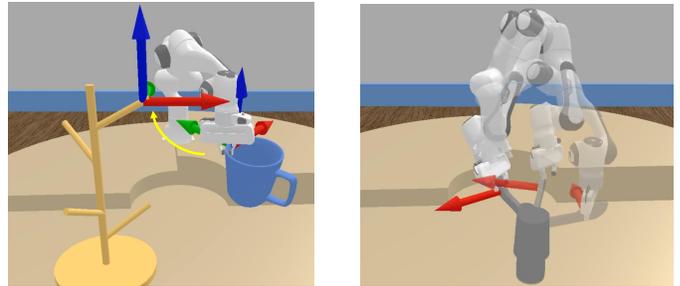
We systematically evaluate RiEMann in both simulation and real-world experiments. This evaluation includes 3 types of manipulation tasks in simulation environments and 2 types of manipulation tasks in the real world, with 5 different settings and 10 demonstrations for each task. We perform quantitative evaluations with success rates SE(3) geodesic distances on RiEMann and comparable baselines, and find RiEMann is consistently superior to other SE(3)-equivariant methods.

A. Simulation Environments and Tasks

We build all simulation manipulation environments based on *sapien* [53] for point-cloud-based manipulation. We build a ring-table experiment platform for all tasks in simulation, as shown in Figure 4. The radius of the table is $0.75m$, and we put a Franka Panda robot arm in the center of the table. We divide the table into a semicircle part and two quarter parts and make them different heights with a height difference of $0.1m$. This is designed to conveniently apply SE(3) transformations on target objects. We put 6 RGBD cameras around the table to fuse their images to get the point cloud input. We cut the input point cloud into a cube with a side length of $2m$ centered on the center of the table. We then downsample the scene point cloud to 8192 points, which is the final input for networks.

For each manipulation task, we collect demonstrations and train our policy under the *training setting* (**T**), and test the trained policy on **T** and four extra settings:

- **New Instance (NI)**: the target object will be replaced by different objects of the same category.



(a) Placing action calculation for the task *Mug on Rack* in simulation. Both the mug pose and rack pose are predicted after picking up the mug.

(b) Opening direction (red arrow) prediction in the task *Turn Faucet*. Note the network can capture the local equivariance of the handle part of the faucet.

Fig. 5: The placing action and opening action.

- **New Poses (NP)**: the initial and target poses of target objects will be under SE(3) transformations on the table.
- **Distracting Objects (DO)**: there will be additional distracting objects in the scene.
- **ALL**: the combination of NI, NP, and DO.

We design three tasks for evaluation. More detailed task descriptions can be found in D. We describe the tasks and demonstrations here:

- 1) *Mug on Rack*: For **T**, we put a mug on the left-down quarter of the table, and put the rack on the right-down quarter of the table, allowing them to rotate within 90 degrees only around the z-axis randomly. For **NI**, we use a new mug and keep the rack the same. For **NP**, we allow the mug and the rack on any quarter of the table, and rotate the mug along all 3 axes with arbitrary degrees.
- 2) *Plane on Shelf*: This task is designed for the evaluation of objects with complex geometry, as evaluated in [54]. The setting of **T**, **NI**, and **NP** are the same as above.
- 3) *Turn Faucet*: This is an articulated object manipulation task and the robot must predict SE(3)-equivariant opening

TABLE I: Success rates of different tasks in simulation. Each value is evaluated under 20 random seeds.

Method	Mug on Rack					Plane on Shelf					Turn Faucet				
	T	NI	NP	DO	ALL	T	NI	NP	DO	ALL	T	NI	NP	DO	ALL
PerAct [46]	0.85	0.00	0.70	0.00	0.00	0.90	0.00	0.80	0.00	0.00	0.45	0.00	0.50	0.00	0.00
R-NDF [48]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	n/a	n/a	n/a	n/a	n/a
EDF [42]	1.00	0.85	1.00	0.95	0.80	0.90	0.75	0.80	0.85	0.70	n/a	n/a	n/a	n/a	n/a
D-EDF [43]	1.00	0.85	0.95	0.95	0.75	1.00	0.80	0.95	0.95	0.75	n/a	n/a	n/a	n/a	n/a
RiEMann (Ours)	1.00	0.90	0.95	1.00	0.85	1.00	0.90	1.00	1.00	0.90	1.00	0.75	1.00	1.00	0.65

TABLE II: SE(3) Geodesic distances of different tasks in simulation. Each value is evaluated under 20 random seeds.

Method	Mug on Rack					Plane on Shelf					Turn Faucet				
	T	NI	NP	DO	ALL	T	NI	NP	DO	ALL	T	NI	NP	DO	ALL
PerAct [46]	0.393	4.086	0.698	4.166	4.375	0.431	4.806	0.469	4.752	4.993	0.457	4.365	0.382	4.218	4.039
R-NDF [48]	4.855	4.298	4.178	4.509	4.662	4.277	4.361	4.179	4.466	4.989	4.996	4.374	4.278	4.229	4.560
EDF [42]	0.249	0.429	0.347	0.252	0.501	0.333	0.872	0.461	0.337	0.985	0.188	1.473	0.448	0.242	2.049
D-EDF [43]	0.312	0.545	0.425	0.337	0.682	0.328	0.966	0.417	0.345	1.024	0.304	2.047	0.567	0.488	2.249
RiEMann (Ours)	0.053	0.066	0.069	0.056	0.068	0.101	0.120	0.117	0.099	0.122	0.079	0.159	0.098	0.082	0.197

	Training Time	Inference Time	Training Batch Size
RiEMann (Ours)	47mins	0.19s	4
D-EDFs [43]	40 mins	15.2s	1

TABLE III: Training time, inference time, and the GPU memory usage (on NVIDIA A40) of RiEMann and D-EDFs on the task *Mug on Rack*. RiEMann has a much faster speed for inference (**5.4FPS**) and lower GPU usage.

direction to turn the faucet. For **NP**, we rotate the faucet along the z -axis, and also change the initial position of the handle. For baselines, since they cannot predict target directions, we only evaluate the quantitative results of their pose estimations.

B. Baselines

We compare RiEMann with four baselines:

- 1) PerAct [46]: a point-cloud-based imitation learning method. This is for the comparison between the SE(3)-equivariant method with the non-SE(3)-equivariant method. We perform SE(3) data augmentation for PerAct.
- 2) R-NDFs [48]: a strong SE(3)-equivariant baseline for manipulation tasks. This represents a line of methods that rely on NDFs [47] and Vector Neurons [11]. We use their pre-trained object encoder from NDFs [47].
- 3) EDFs [42]: an SE(3)-equivariant baseline that use SE(3)-transformers for manipulation tasks. This is for the comparison between directly regressing target poses (RiEMann) and field matching on type- l fields. Note EDFs require separated point clouds of the scene and the robot end-effector respectively. Here we manually extract the end-effector point cloud out from the scene for EDFs.
- 4) D-EDFs [43]: the state-of-the-art SE(3)-equivariant baseline for object rearrangement tasks. This is for the comparison of training and inference time, since D-EDFs claim their main advantage is training speed.

C. Evaluation Metrics and Methodology

We first report the success rates of all methods in all five settings (**T**, **NI**, **NP**, **DO**, **ALL**) with the average success rates across 20 different random seeds. The success rates are the total success rates of the two stages of each task.

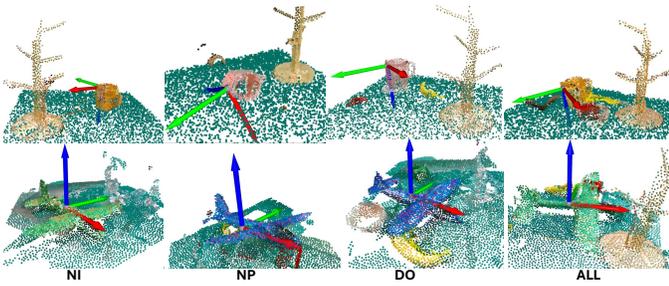
However, we point out that only using success rates cannot effectively reflect the effect of different methods, because the success of the task not only relies on the accurate prediction of the target poses, but also relies on different object shapes and specific task settings. For example, the size of the inner radius of the mug handle can greatly affect the success rates, since a larger handle is easier for hanging. To quantitatively show the model’s effects and eliminate the influences of other factors, we propose to evaluate the SE(3) geodesic distance [5] of the predicted target pose $\hat{\mathbf{T}}$ and the ground truth \mathbf{T} :

$$\mathcal{D}_{geo}(\mathbf{T}, \hat{\mathbf{T}}) = \sqrt{\left\| \log \left(\mathbf{R}^\top \hat{\mathbf{R}} \right)^\vee \right\|^2 + \|\hat{\mathbf{t}} - \mathbf{t}\|^2}, \quad (6)$$

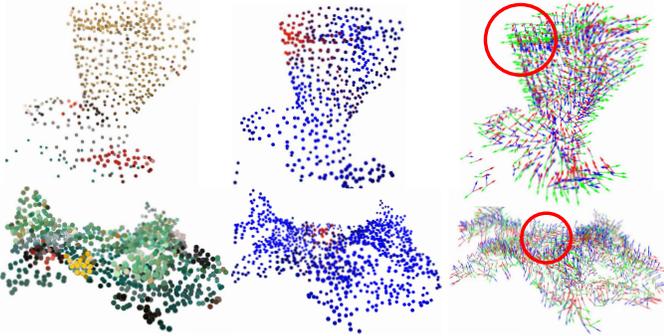
where \vee is the logmap. We report \mathcal{D}_{geo} in the same manner as success rates.

D. Results

Table I and Table II show the success rates and \mathcal{D}_{geo} respectively. We can see that PerAct [46] generally performs worse than SE(3)-equivariant methods, which shows the advantage of improving the sample efficiency of equivariant methods. NDFs [48] with Vector Neurons [11] totally fail on unsegmented point cloud input, which shows the necessity of using local SE(3)-equivariance modules. EDFs [42] and D-EDFs [43] are generally similar in both success rate and \mathcal{D}_{geo} , and can generalize to local SE(3) transformations of the target objects. However, their performances are not satisfactory because of their field-matching process which makes the task a harder learning problem. Meanwhile, they cannot output any other physical quantities such as the direction of turning the faucet. Instead, RiEMann consistently achieves the best



(a) Pose predictions of the task *Mug on Rack* and *Plan on Box* of four *test* cases **NI**, **NP**, **DO**, and **ALL** in the real world.



(b) The B_{ROI} and the local $SE(3)$ -equivariant feature visualization of the above **ALL** test cases. The middle shows the position (one type-0 vector) field predicted from ϕ_1 . The right shows the orientation (three type-1 vectors) fields from ϕ_2 . We can see that the type-1 vectors away from the target position are different from the correct orientation, which shows the necessity of r_2 shown with the red circle.

Fig. 6: Test pose predictions and feature visualization of real-world evaluations.

results in both training and testing settings, which shows the superiority of our end-to-end learning paradigm. Note a $\mathcal{D}_{geo} = 0.05$ means an error around $1cm + 2^\circ$, which is low enough for a successful manipulation. We also visualize the four test case pose predictions in Figure 6a and the local $SE(3)$ -equivariance features in Figure 6b. We can see that RiEMann can generalize to local $SE(3)$ -transformations of target objects and new instances, and resist the distraction of other objects.

We also evaluate the training and inference time of RiEMann and D-EDFs [43], as well as their GPU memory usage, as shown in Table III. RiEMann is the only method that can achieve fast training and near real-time inference as well as local $SE(3)$ -equivariance.

E. Ablation Studies

We also conduct ablation studies for further analysis of key components of RiEMann that may influence the performance, which include: 1) Point numbers in the point cloud; 2) Different maximum types of descriptor vectors in the hidden layer (four layers for each network); 3) The number of demonstrations per task; 4) Using and not using the saliency map network ϕ . We test the trained model on the **NP** case of the task *Mug on Rack* in simulation. Results are shown in Figure 7. We can see that with more points in the scene input point cloud, the performance of the model is better, and the

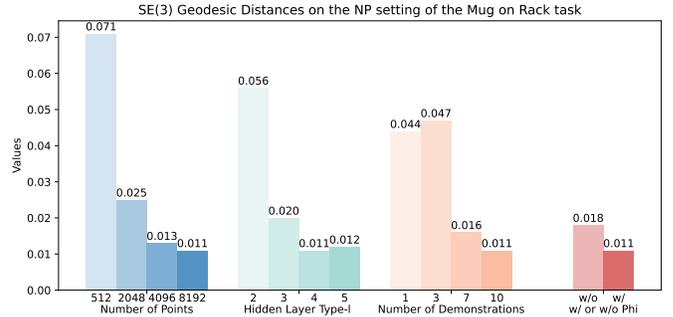


Fig. 7: Ablation studies of different hyperparameters of RiEMann. Each value is the average result of 20 random seeds.

Task	T		NI		NP		DO		ALL	
	G	A	G	A	G	A	G	A	G	A
Mug on Rack	1.0	1.0	1.0	1.0	0.75	0.75	0.92	0.83	0.75	0.58
Plane on Shelf	1.0	1.0	1.0	1.0	0.58	0.50	1.0	1.0	0.55	0.50

TABLE IV: Success rates of *Mug on Rack* and *Plane on Shelf* in the real world. Each value is the average of 12 tests.

same is the number of demonstrations. Our model can achieve competitive results with less than 10 demonstrations. For the hidden layer type- l experiment, we can see that with a higher max type- l , the model can work better. However, in practice, higher type- l will extremely increase the computational cost and GPU memory usage. In the *Mug On Rack* task, a network with a maximum number l equals 5 only supports batch size = 1 during training on an Nvidia A40. Lastly, the last group of Figure 7 shows that the saliency map network can not only reduce the training burden of the policy network but also improve the final pose estimation results.

F. Real World Evaluation

Finally, we evaluate RiEMann in two real-world experiments: *Mug on Rack* and *Plane on Shelf*, as illustrated in Figure 4. We use four RealSense D435i cameras for point cloud fusion and a Franka Panda arm for execution. We record 10 demonstrations for each task. We show quantitative results of success rates of pick (P) and full task (A) respectively of each task in Table IV. We can see that the real-world performance of RiEMann is generally the same within the simulation, which shows RiEMann can resist the relatively low-quality point cloud input from the real world. The performance of **NP** is notably suboptimal, primarily due to the partial absence of the object’s point cloud in real-world scenarios. This deficiency stems from the cameras’ inability to capture the lower section of the object, resulting in a missing portion that faces downwards. Consequently, in both **T** and **NP**, these specific parts exhibit noticeable disparities, as illustrated in Figure 8. Please also check supplementary videos for more results and the real-time following video.

VI. DISCUSSION AND FUTURE WORKS

This paper proposes RiEMann, a near real-time $SE(3)$ -equivariant robot manipulation framework without point cloud

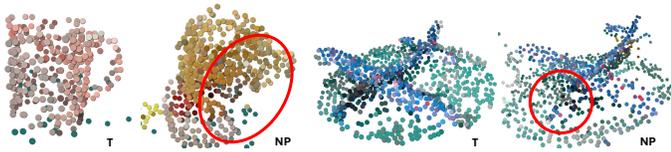


Fig. 8: Point cloud visualization of the **T** case and the **NP** case in the real world. Different parts of target objects are missing (marked with the red circle), which causes the poor performance of **NP**.

segmentation. We achieve this by our specially designed parameterization of the action space and training scheme that makes RiEMann a supervised model, and achieves superior SE(3) generalization ability in experimental results on both simulation and real-world manipulation tasks. One limitation is that equivariant models usually bring more computation and memory costs, which hinders their application in large-scale robot learning scenarios. Another limitation is that RiEMann does not perform well for occluded point cloud input or symmetric objects. Future works can focus on these two problems, and seek to apply RiEMann on other tasks that require SE(3)-equivariant outputs such as forces, and seek to incorporate RiEMann into robot reinforcement learning.

ACKNOWLEDGMENTS

We would like to thank Tianren Zhang, Yuanchen Ju, and Chenrui Tie for their helpful discussions.

REFERENCES

- [1] Brandon Anderson, Truong Son Hy, and Risi Kondor. Cormorant: Covariant molecular neural networks. *Advances in neural information processing systems*, 32, 2019.
- [2] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.
- [3] Erik J. Bekkers. An introduction to equivariant convolutional neural networks for continuous groups. <https://uvagedl.github.io/GroupConvLectureNotes.pdf>, 2021.
- [4] Alexander Bogatskiy, Brandon Anderson, Jan Offermann, Marwah Roussi, David Miller, and Risi Kondor. Lorentz group equivariant neural network for particle physics. In *International Conference on Machine Learning*, pages 992–1002. PMLR, 2020.
- [5] Luca Carlone. Lecture 4: Lie groups. Lecture Notes of Visual Navigation for Autonomous Vehicles (VNAV), 2023.
- [6] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.
- [7] Ethan Chun, Yilun Du, Anthony Simeonov, Tomas Lozano-Perez, and Leslie Kaelbling. Local neural de-

scriptor fields: Locally conditioned object representations for manipulation. *arXiv preprint arXiv:2302.03573*, 2023.

- [8] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.
- [9] Taco S Cohen and Max Welling. Steerable cnns. *arXiv preprint arXiv:1612.08498*, 2016.
- [10] Taco S Cohen, Mario Geiger, and Maurice Weiler. A general theory of equivariant cnns on homogeneous spaces. *Advances in neural information processing systems*, 32, 2019.
- [11] Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas J Guibas. Vector neurons: A general framework for so(3)-equivariant networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12200–12209, 2021.
- [12] Carlos Esteves. Theoretical aspects of group equivariant neural networks. *arXiv preprint arXiv:2004.05154*, 2020.
- [13] Carlos Esteves, Christine Allen-Blanchette, Xiaowei Zhou, and Kostas Daniilidis. Polar transformer networks. *arXiv preprint arXiv:1709.01889*, 2017.
- [14] Marc Finzi, Samuel Stanton, Pavel Izmailov, and Andrew Gordon Wilson. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In *International Conference on Machine Learning*, pages 3165–3176. PMLR, 2020.
- [15] Marc Finzi, Samuel Stanton, Pavel Izmailov, and Andrew Gordon Wilson. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In *International Conference on Machine Learning*, pages 3165–3176. PMLR, 2020.
- [16] Marc Finzi, Max Welling, and Andrew Gordon Wilson. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. In *International conference on machine learning*, pages 3318–3328. PMLR, 2021.
- [17] Peter R Florence, Lucas Manuelli, and Russ Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. *arXiv preprint arXiv:1806.08756*, 2018.
- [18] Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. Se(3)-transformers: 3d roto-translation equivariant attention networks. *Advances in neural information processing systems*, 33:1970–1981, 2020.
- [19] Johannes Gasteiger, Florian Becker, and Stephan Günnemann. Gemnet: Universal directional graph neural networks for molecules. *Advances in Neural Information Processing Systems*, 34:6790–6802, 2021.
- [20] Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, et al. Maniskill2: A unified benchmark for generalizable manipulation skills. *arXiv preprint arXiv:2302.04659*, 2023.
- [21] Jiaqi Han, Yu Rong, Tingyang Xu, and Wenbing Huang.

- Geometrically equivariant graph neural networks: A survey. *arXiv preprint arXiv:2202.07230*, 2022.
- [22] Jiaqi Han, Yu Rong, Tingyang Xu, and Wenbing Huang. Geometrically equivariant graph neural networks: A survey. *arXiv preprint arXiv:2202.07230*, 2022.
- [23] haosulab. Mplib: a lightweight python package for motion planning, 2023. URL <https://github.com/haosulab/MPLib>. GitHub repository.
- [24] Walter Hoffmann. Iterative algorithmen für die gram-schmidt-orthogonalisierung. *Computing*, 41:335–348, 1989.
- [25] Haojie Huang, Dian Wang, Robin Walters, and Robert Platt. Equivariant transporter network. *arXiv preprint arXiv:2202.09400*, 2022.
- [26] Haojie Huang, Dian Wang, Xupeng Zhu, Robin Walters, and Robert Platt. Edge grasp network: A graph-based se (3)-invariant approach to grasp detection. *arXiv preprint arXiv:2211.00191*, 2022.
- [27] Haojie Huang, Dian Wang, Arsh Tangri, Robin Walters, and Robert Platt. Leveraging symmetries in pick and place. *The International Journal of Robotics Research*, page 02783649231225775, 2024.
- [28] Michael J Hutchinson, Charline Le Lan, Sheheryar Zaidi, Emilien Dupont, Yee Whye Teh, and Hyunjik Kim. Lietransformer: Equivariant self-attention for lie groups. In *International Conference on Machine Learning*, pages 4533–4543. PMLR, 2021.
- [29] David Inkyu Kim and Gaurav S Sukhatme. Semantic labeling of 3d point clouds with object affordance for robot manipulation. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5578–5584. IEEE, 2014.
- [30] Seungyeon Kim, Byeongdo Lim, Yonghyeon Lee, and Frank C Park. Se (2)-equivariant pushing dynamics models for tabletop object manipulations. In *Conference on Robot Learning*, pages 427–436. PMLR, 2023.
- [31] Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.
- [32] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pages 5639–5650. PMLR, 2020.
- [33] Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33:19884–19895, 2020.
- [34] Jiahui Lei, Congyue Deng, Karl Schmeckpeper, Leonidas Guibas, and Kostas Daniilidis. Efem: Equivariant neural field expectation maximization for 3d object segmentation without scene supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4902–4912, 2023.
- [35] Yi-Lun Liao and Tess Smidt. Equiformer: Equivariant graph attention transformer for 3d atomistic graphs. *arXiv preprint arXiv:2206.11990*, 2022.
- [36] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *arXiv preprint arXiv:2306.03310*, 2023.
- [37] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021.
- [38] Lucas Manuelli, Wei Gao, Peter Florence, and Russ Tedrake. kpm: Keypoint affordances for category-level robotic manipulation. In *The International Symposium of Robotics Research*, pages 132–157. Springer, 2019.
- [39] Diego Marcos, Michele Volpi, Nikos Komodakis, and Devis Tuia. Rotation equivariant vector field networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5048–5057, 2017.
- [40] Alexandre Milesi. Se(3)-transformers for pytorch, 2021. URL <https://github.com/NVIDIA/DeepLearningExamples/tree/master/DGLPyTorch/DrugDiscovery/SE3Transformer>.
- [41] Harish Ravichandar, Athanasios S Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. *Annual review of control, robotics, and autonomous systems*, 3:297–330, 2020.
- [42] Hyunwoo Ryu, Hong-in Lee, Jeong-Hoon Lee, and Jongeun Choi. Equivariant descriptor fields: Se (3)-equivariant energy-based models for end-to-end visual robotic manipulation learning. *arXiv preprint arXiv:2206.08321*, 2022.
- [43] Hyunwoo Ryu, Jiwoo Kim, Junwoo Chang, Hyun Seok Ahn, Joohwan Seo, Taehan Kim, Jongeun Choi, and Roberto Horowitz. Diffusion-edfs: Bi-equivariant denoising generative modeling on se (3) for visual robotic manipulation. *arXiv preprint arXiv:2309.02685*, 2023.
- [44] Daniel Seita, Pete Florence, Jonathan Tompson, Erwin Coumans, Vikas Sindhwani, Ken Goldberg, and Andy Zeng. Learning to rearrange deformable cables, fabrics, and bags with goal-conditioned transporter networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4568–4575. IEEE, 2021.
- [45] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1134–1141. IEEE, 2018.
- [46] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023.
- [47] Anthony Simeonov, Yilun Du, Andrea Tagliasacchi, Joshua B Tenenbaum, Alberto Rodriguez, Pulkit Agrawal, and Vincent Sitzmann. Neural descriptor fields:

- Se (3)-equivariant object representations for manipulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6394–6400. IEEE, 2022.
- [48] Anthony Simeonov, Yilun Du, Yen-Chen Lin, Alberto Rodriguez Garcia, Leslie Pack Kaelbling, Tomás Lozano-Pérez, and Pulkit Agrawal. Se (3)-equivariant relational rearrangement with neural descriptor fields. In *Conference on Robot Learning*, pages 835–846. PMLR, 2023.
- [49] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.
- [50] Dian Wang, Mingxi Jia, Xupeng Zhu, Robin Walters, and Robert Platt. On-robot learning with equivariant models. *arXiv preprint arXiv:2203.04923*, 2022.
- [51] Dian Wang, Robin Walters, and Robert Platt. So (2)-equivariant reinforcement learning. In *International Conference on Learning Representations*, 2022.
- [52] Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco S Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. *Advances in Neural Information Processing Systems*, 31, 2018.
- [53] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A simulated part-based interactive environment. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [54] Zhengrong Xue, Zhecheng Yuan, Jiashun Wang, Xueqian Wang, Yang Gao, and Huazhe Xu. Useek: Unsupervised se (3)-equivariant 3d keypoints for generalizable manipulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1715–1722. IEEE, 2023.
- [55] Jingyun Yang, Congyue Deng, Jimmy Wu, Rika Antonova, Leonidas Guibas, and Jeannette Bohg. Equivact: Sim (3)-equivariant visuomotor policies beyond rigid object manipulation. *arXiv preprint arXiv:2310.16050*, 2023.
- [56] Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, et al. Transporter networks: Rearranging the visual world for robotic manipulation. In *Conference on Robot Learning*, pages 726–747. PMLR, 2021.
- [57] Xupeng Zhu, Dian Wang, Ondrej Biza, Guanang Su, Robin Walters, and Robert Platt. Sample efficient grasp learning using equivariant models. *arXiv preprint arXiv:2202.09468*, 2022.

A. Iterative Modified Gram-Schmidt Orthogonalization

We use Iterative Modified Gram-Schmidt Orthogonalization [24] to make the outputted rotation matrix $\hat{\mathbf{R}}$ legal. IMGS works much more stable than the vanilla Gram-Schmidt Orthogonalization. The algorithm is summarized as follows.

Algorithm 2 Iterative Modified Gram-Schmidt

Input: $\hat{\mathbf{R}}$ that contains column vectors $v_0, v_1,$ and $v_2 \in \mathbb{R}^3$

```

1: for iter = 1 to 2 do
2:   for i = 0 to 2 do
3:      $u_i = v_i$ 
4:     for j = 0 to i - 1 do
5:        $v_i = v_i - \frac{\langle v_i, u_j \rangle}{\langle u_j, u_j \rangle} u_j$ 
6:     end for
7:      $u_i = v_i$ 
8:   end for
9: end for

```

Output: A legal rotation matrix $\hat{\mathbf{R}}$ that contains updated column vectors $v_0, v_1,$ and $v_2 \in \mathbb{R}^3$

B. Proofs

First, let's review the definition SE(3)-equivariance on our point cloud \mathbf{P} . Given an outputted vector field $\mathbf{f}_{out}(x) = \bigoplus_{i=1}^n \mathbf{f}^i(x), \forall x \in \mathbf{P}$ from SE(3)-transformer [18] where n is the total types of vectors, they are SE(3)-equivariant that means:

$$\mathbf{D}_l(\mathbf{R})\mathbf{f}_l(x) = \mathbf{f}_l(Tx), \forall x \in \mathbf{P}, T = (\mathbf{R}, \mathbf{t}) \in SE(3), l \in n. \quad (7)$$

where $\mathbf{D}_l(R)$ is the Wigner-D matrix. For entities living in the usual 3D physical world, the angular momentum quantum number j of the Wigner-D matrix is 1, thus for vectors with $l = 1$, we have:

$$\mathbf{D}(\mathbf{R}) = e^{-im'\alpha} d_{m',m}^j(\beta) e^{-im\gamma}, \quad (8)$$

where α, β, γ are the Euler angle representation of \mathbf{R} that satisfies $\mathbf{R} = \mathbf{R}_z(\alpha)\mathbf{R}_x(\beta)\mathbf{R}_z(\gamma)$, $m, m' \in \{-1, 0, 1\}$, and $d_{m',m}^j(\beta)$ is the matrix element. Since $\mathbf{D}(\mathbf{R})$ is also a unitary matrix, we can find a set of basis $[v_0, v_1, v_2]$ that makes $\mathbf{D}(\mathbf{R}) = \mathbf{R}$.

For type-0 vector fields, $\mathbf{D}_0(\mathbf{R})$ is a one-dimensional identical scale factor 1. Let's begin our proofs.

Theorem 1 *Rotation matrices, represented by three type-1 vectors, are SE(3)-equivariant parameterization of rotation actions.*

Proof: For a rotation matrix $\mathbf{R} = [v_0, v_1, v_2] \in \mathbb{R}^9$, where $v_0, v_1,$ and v_2 are the three column vectors, we use three type-1 vectors to represent these three column vectors. Thus the output vector of the network is as follows:

$$\mathbf{f}_{out}(x) = \bigoplus_{i=1}^3 \mathbf{f}_i^1(x), \forall x \in \mathbf{P}. \quad (9)$$

When the input point cloud is transformed by a SE(3) transformation $\mathbf{T} = (\mathbf{R}, \mathbf{t})$, the rotation matrix representation of the target object also transforms by \mathbf{R} , that is $[v'_0, v'_1, v'_2] = \mathbf{R}[v_0, v_1, v_2]$. According to Equation 7 and 8, the outputted type-1 vector fields are transformed by $\mathbf{D}_1(\mathbf{R}) = \mathbf{R}$, thus we have:

$$[v'_0, v'_1, v'_2] = \mathbf{D}_1(\mathbf{R})[v_0, v_1, v_2] \quad (10)$$

Theorem 2 *There is no SE(3)-equivariant vector field representation for Euler angle, quaternion, and axis-angle.*

Proof: We use proof by contradiction to prove theorem 2. Consider the example illustrated in Figure 2.

1) For quaternion, we define a quaternion $q = [\cos(\theta/2), \sin(\theta/2)u_i, \sin(\theta/2)u_j, \sin(\theta/2)u_k]$ where θ is the rotation angle and $[u_i, u_j, u_k]$ is the rotation axis. For the initial pose, we have $q = [1, 0, 0, 0]$. For the end pose, we have $q' = [\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, 0, 0]$, thus:

$$q' = q + [\frac{\sqrt{2}}{2} - 1, \frac{\sqrt{2}}{2}, 0, 0]. \quad (11)$$

There are two options for quaternion type- l parameterization: 1) using four type-0 vectors; 2) using one type-1 vector and one type-0 vector. For both cases, the $+\frac{\sqrt{2}}{2} - 1, \frac{\sqrt{2}}{2}, 0, 0]$ operation cannot be represented by a Wigner-D matrix. Thus there is no SE(3)-equivariant vector field representation for quaternion. Axis-angle can be proven in the same way.

2) For Euler angles, we define an Euler angle as $E = (\alpha, \beta, \gamma)$. For the initial pose, we have Euler angles equal to $E = (0, 0, 0)$. For the end pose, we have Euler angles equal to $E' = (-\frac{\pi}{2}, 0, 0)$. Thus we have:

$$E' = E + [-\frac{\pi}{2}, 0, 0]. \quad (12)$$

There are two options for Euler angles' type- l parameterization: 1) using three type-0 vectors; 2) using one type-1 vector. For both cases, the $+\frac{\pi}{2}, 0, 0]$ operation cannot be represented by a Wigner-D matrix. Thus there is no SE(3)-equivariant vector field representation for Euler angles. ■

C. Training Details

1) *Point Cloud Preprocessing:* In the real-world experiments, we perform point cloud voxel downsampling before feeding the point cloud into the network with the voxel size equal to 1cm for the *Mug on Rack* task and 2cm for the task *Plane on Shelf*.

After this, we perform color jittering by adding Gaussian noise on each point's color with a standard variance equal to 0.005. We also perform random color dropping that replaces 30% points' color to zero, and HSV transformation that randomly transfers the hue, saturation, and brightness of each point by 0.4, 1.5, and 2 times respectively.

2) *Method Details*: For RiEMann, We use $r_1 = 0.2m, r_2 = 0.02m$ for all the tasks in the simulation. We use $r_1 = 0.16m, r_2 = 0.02m$ for the real world *Mug on Rack* task, and $r_1 = 0.2m, r_2 = 0.02m$ for the *Plane on Shelf* task. Other network hyperparameters of RiEMann are listed in Table V. We do not use any kind of prior knowledge for the training of RiEMann such as object segmentation, pertaining, or pose augmentations. We also set the robot to first reach some pre-defined pre-grasp pose (e.g., above the mug) and a pre-place pose (e.g., in front of the rack) to eliminate the unnecessary influence of the motion planners.

The full training of RiEMann takes 200 epochs on a single NVIDIA A40 with a batch size of 4 and a learning rate of $1e-4$ for each network module ϕ , ψ_1 , and ψ_2 . However, we find that for *Mug on Rack*, they only need about 50 epochs to converge, which takes about 47 minutes.

	Network Layer	Max Type- l	Head Number	Channels	Message Passing Distance
ϕ	4	4	1	8	0.1m
ψ_1	4	3	1	8	0.07m
ψ_2	4	4	1	8	0.07m

TABLE V: Network hyperparameters of RiEMann.

For PerAct [46], the language descriptions of our tasks are: *Put the mug on the rack*, *Place the plane on the shelf*, *Turn the faucet*. We follow the original 3D voxel grid size (100^3) and the patch size (5^3). We use Euler angles as the rotational action representations for PerAct. For fairness, we do not train the gripper action for PerAct. We use 6 self-attention layers for the perceiver Transformer module. The other hyper-parameters are the same with the original paper.

For R-NDF [48], since there is no pre-trained weight for the plane and the faucet, we here pre-train the NDFs using the reconstructed meshes from the point clouds in our demonstrations, and use these model as the NDFs module to run R-NDFs. We observe that R-NDFs fail to accomplish all of the tasks when testing, which shows that R-NDFs cannot perform well without object segmentation, because of the locality requirements of R-NDFs. We also tried to use the original pre-trained weights from the original paper [48] for the task *Mug on Rack*, but we found that the performances were even worse because of the discrepancy of the specific object shapes in the test experiments and the pertaining datasets. Other network hyper-parameters are the same as in the original paper.

For EDF [42] and D-EDF [43], we manually separate the robot end-effector and the grasped object point cloud from the scene rather than setting a series of separate cameras to capture their point cloud. For EDFs, we run the MH for 1000 steps, run the Langevin algorithm for 300 steps, and optimize the samples for 100 steps. We use one query point for picking and three query points for placing. We train EDFs for 200 epochs, the same epochs with RiEMann. For D-EDFs, we train the networks for 1 hour with parallel training of the low-resolution and the high-resolution networks and the energy-based critic network. Other network hyper-parameters are the same as in the original paper.

D. Simulation Experiments

1) *Detailed Descriptions of Simulation Tasks*: *Mug on Rack*: A mug and a rack are placed on the table. The robot has to pick up the mug by the rim and then hang it on the rack by the handle. This is the most representative object rearranging task that is also evaluated in [47, 42, 48, 43]. For the training set \mathbf{T} , we use a mug in blue with a side length of about 17 cm, and a rack with a height of 67cm. The mug must be hung on the highest peg of the rack. For the new instance set \mathbf{NI} , we use a patterned red mug with a height of about 19cm and a base diameter of about 10cm.

Plane on Shelf: A plane model and a box-shape shelf are placed on the table, and the robot has to pick the middle part of the body of the plane and place it on the shelf. For \mathbf{T} , we use a grey plane model with a length of about 20cm. For \mathbf{NI} , we use a blue plane with the same size.

Turn Faucet: A faucet is placed on the table, and the robot has to turn on the faucet by first moving to the handle of the faucet and then moving along the opening direction. For \mathbf{T} , we use the NO. 5004 faucet model in ManiSkill2 [20]. For \mathbf{NI} , we use the No. 5005 faucet model.

2) *Demonstration Collection*: We provide the ground truth pose to a point cloud based motion planner MPLib [23] to generate the demonstration trajectory for training. We transform all point cloud input to the end-effector coordinate system. We collect 10 demonstrations for each setting for the evaluation of SE(3) geodesic distance. We manually exclude those situations that cannot support a successful collision-free motion planning trajectory, as well as in the testing cases.

E. Real World Experiments

1) *Environment Setup*: We use a Franka Emika Panda robot arm with four RealSense D435i RGB-D cameras for the real-world experiments, as shown in Figure 4. The cameras are calibrated relative to the robot’s base frame. We fuse the point clouds from all four cameras and transform the point cloud into the end-effector frame of the robot for control. We crop the scene to a cube with a side length of 1.5 meters and downsample the point to get 8192 points in the scene.

For real-world tasks, since the point cloud is noisy and usually part-occluded, we do not perform the pose transformation calculation $\hat{\mathbf{T}} = \mathbf{T}_{place} \mathbf{T}_{object}^{-1}$ for the task, i.e., we directly use the predicted target pose as the final action.

2) *Detailed Task Descriptions*: *Mug on Rack*: The task is similar to the version in the simulation. For \mathbf{T} , we use a pink mug with a side length of about 10cm. We use a rack with a height of 35cm and a base diameter of about 15cm. We split the table into four equal areas, as in the simulated version of this task. In \mathbf{T} , we only collect demonstrations in a quarter of the desktop area and let the mug rotate along the z-axis for 90 degrees in a top pose. For \mathbf{NI} , we use a yellow new mug with a similar size to the pink mug. For \mathbf{NP} , we let the mug on all table regions and rotate in 3 dimensional with any degree.

Plan on Shelf: The task is similar to the version in the simulation. For \mathbf{T} , we use a blue plane model, and collect demonstrations in the same manner as above. For \mathbf{NI} , we use

a green plane with a similar size. The shelf is a set of discrete racks that can support the plane if it is placed in the correct pose.

3) *Demonstration Collection:* We use the teaching mode of the robot arm to give demonstrations, as illustrated in Figure 1 and in supplementary videos. We transform all the point cloud into the end-effector coordinate system.