EquivAct: SIM(3)-Equivariant Visuomotor Policies beyond Rigid Object Manipulation

Jingyun Yang*[1], Congyue Deng*[1], Jimmy Wu[2], Rika Antonova[1], Leonidas Guibas[1], Jeannette Bohg[1]

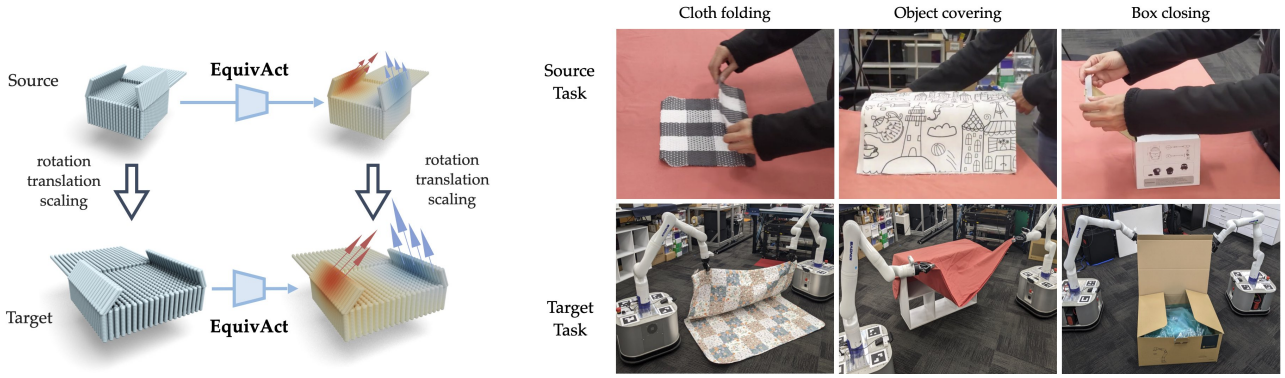

Fig. 1: **Overview.** Constructed with SIM(3)-equivariant point cloud networks, our method takes a few examples of solving a source task, then generalizes zero-shot to changes in object *appearances*, *scales*, and *poses*.

***Abstract*— If a robot masters folding a kitchen towel, we would also expect it to master folding a beach towel. However, existing works for policy learning that rely on data set augmentations are still limited in achieving this level of generalization. Our insight is to add equivariance to both the visual object representation and policy architecture. We propose *EquivAct* which utilizes SIM(3)-equivariant network structures that guarantee generalization across all possible object translations, 3D rotations, and scales by construction. Training of EquivAct is done in two phases. We first pre-train a SIM(3)-equivariant visual representation on simulated scene point clouds. Then, we learn a SIM(3)-equivariant visuomotor policy on top of the pre-trained visual representation using a small amount of source task demonstrations. We demonstrate that after training, the learned policy directly transfers to objects that substantially differ in scale, position and orientation from the source demonstrations. In simulation, we evaluate our method in three manipulation tasks involving deformable and articulated objects thereby going beyond the typical rigid object manipulation tasks that prior works considered. We show that our method outperforms prior works that do not use equivariant architectures or do not use our contrastive pre-training procedure. We also show quantitative and qualitative experiments on three real robot tasks, where the robot watches twenty demonstrations of a tabletop task and transfers zero-shot to a mobile manipulation task in a much larger setup. Project website: https://equivact.github.io**

I. Introduction

Given a few examples of how to solve a manipulation task, humans can extrapolate and learn to solve variations of the same task where the objects involved have different visual or physical properties. Existing works in robot learning still require extensive data augmentation to make the learned policies generalize to varied object scales, orientations, and visual appearances [1]–[5]. Even then, augmentations do not guarantee generalization to unseen variations.

In this work, we focus on the problem of learning a visuomotor policy that can take a few example trajectories from a single source manipulation scenario as input and generalize zero-shot to scenarios with unseen object visual appearances, scales, and poses. In particular, we want the learned policies to be capable beyond mere pick-and-place of rigid objects and also handle deformable and articulated objects, such as clothes or boxes. Our insight is to *add equivariance into both the visual object representation and policy architecture*, so that the learned policy generalizes to novel object positions, orientations, and scales *by construction*.

We present *EquivAct*, a novel visuomotor policy learning method that can learn closed-loop policies for 3D robot manipulation tasks from demonstrations in a single source manipulation scenario and generalize zero-shot to unseen scenarios. The learned policy takes a partial point cloud of the scene and the robot end-effector poses as input and outputs robot actions, which include end-effector velocity and gripper commands. Different from the neural network architecture that most prior work uses, we employ SIM(3)-equivariant network structures. This means that when the input point cloud and end-effector positions are translated and rotated, the output end-effector velocities will transform accordingly. Equipped with this equivariance, our policy architecture can learn from demonstrations of smaller-scale tabletop tasks, and then zero-shot generalize to mobile manipulation tasks that involve larger variants of the demonstrated objects with different visual and physical appearances.

Our method is composed of two phases: a representation

* Equal contribution. [1] Department of Computer Science, Stanford University. [2] Department of Computer Science, Princeton University.

This work was supported in part by the Toyota Research Institute and the National Science Foundation grant No.2030859 to the Computing Research Association for the CIFellows Project.

Contact: {jingyuny, rika.antonova}@stanford.edu

learning phase and a policy learning phase. In the representation learning phase, the agent is given a set of simulated point clouds that are recorded from objects of the same category as the objects in the target task but with a randomized non-uniform scaling. While the proposed architecture is equivariant to uniform scaling, we need to augment the training data in this way to account for non-uniform scaling. Note that the simulated data is not a demonstration of the target task and does not need to include robot actions. With the simulated data, we train a SIM(3)-equivariant [6] encoder-decoder architecture that takes the scene point cloud as input and outputs the global and local features of the input point cloud. We use a contrastive learning loss on paired point cloud inputs during training, so that local features for corresponding object parts of objects in similar poses can be pulled closer than non-corresponding parts. In the policy learning phase, we assume access to a small set of demonstrated trajectories of the task. This could be in the form of human demonstrations or teleoperated robot trajectories. With the demonstration data, we train a closed-loop policy that takes a partial point cloud of the scene as input, uses the pre-trained encoder from the representation learning phase to obtain the global and local features of the input point cloud, and then passes the features through a SIM(3)-equivariant action prediction network to predict end-effector movements.

We evaluate our method in three challenging tasks that go beyond the typical rigid object manipulation tasks of prior work [7]–[11]: comforter folding, container covering, and box closing (see Fig. 1). In each task, we provide a small amount of human demonstrations where a human operates a small object on a table. Then, we evaluate the approach on a mobile manipulation platform, where the robots need to solve a similar task on a much larger scale than what is shown in the demonstrations. We show that our approach can successfully learn a closed-loop robot manipulation policy from the source manipulation demonstrations, and complete the target task zero-shot without any further adaptation. We also show that our method outperforms works that do not leverage equivariance and is more efficient than methods that rely on extensive augmentations for generalization to out-of-distribution object poses and scales.

II. Related Work

A. Equivariant 3D Learning

A number of 3D deep learning works have studied the construction of neural network architectures that are equivariant to 3D transformations such as rigid transformations [12]–[19], multi-part motions [6], [20]–[22], or more general and arbitrary group actions [23]–[27]. Such network designs enable natural and robust generalizations to out-of-distribution inputs by construction without additional data augmentation in the training process. In our work, we leverage the implicit shape representation from [6] powered by the network structures from [15] to facilitate more geometrically interpretable and generalizable feature learning.

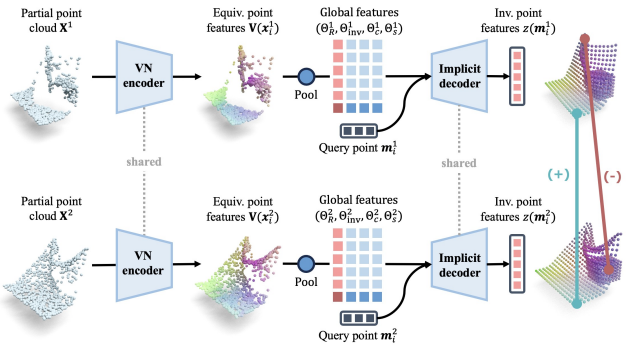

Fig. 2: **Representation learning pipeline.** Our representation takes a partial point cloud as input and outputs equivariant global and local features of the point cloud. To train this representation, the algorithm takes paired partial point clouds as inputs, processes them through an equivariant encoder-decoder architecture, and computes a contrastive loss based on invariant point features, which are used to update the learned representation.

B. Equivariant Representations for Robot Manipulation

Prior works have studied equivariant features for robot manipulation [7]–[11]. But these works have a series of limitations: (1) they are designed to handle only pick-and-place-like tasks, and are not capable of learning harder tasks involving articulated and deformable objects; (2) they are mostly based on open-loop policies or hand-designed pick-and-place primitives, so their framework does not support closed-loop feedback, which is important for scenarios that involve deformable and articulated objects, where the policy needs to react to changes in object poses throughout task completion; (3) they only handle equivariances in translation and rotation and did not consider scale equivariance. [28] learns an SO(2) equivariant closed-loop manipulation policy using additional simulated augmentations in a suite of tabletop rigid object manipulation tasks, but cannot handle out-of-distribution objects with unseen uniform and non-uniform scaling. Compared to prior works, our work learns a robot manipulation policy that supports closed-loop feedback, can handle deformable and articulated objects, is equivariant to translation, rotation, and uniform scaling by construction, and achieves generalization for objects with non-uniform scaling with a representation learning phase.

C. 3D Representations for Deformable and Articulated Object Manipulation

The problem of 3D manipulation of deformable and articulated objects has been studied by a number of prior works, including FlingBot [29], GarmentNets [30], FabricFlowNet [31], and ACID [32]. However, these works often are very focused on specific manipulation problems. For example, FlingBot focuses on the task of flinging only, FabricFlowNet is designed for pick-and-place tasks on cloths, and ACID is designed to manipulate volumetric deformable objects. We design a framework that is capable of learning a variety of 3D manipulation tasks, including ones that involve deformable and articulated objects.

III. Method

A. Preliminaries

SIM(3)-equivariance. Given a function f which takes a point cloud $\mathbf{X} \in \mathbb{R}^{N\times 3}$ as input, we say it is $\mathrm{SIM}(3)$-equivariant if for any 3D rigid transformation $\mathbf{T} = (\mathbf{R}, \mathbf{t}, s) \in \mathrm{SIM}(3)$ with rotation $\mathbf{R}$, translation $\mathbf{t}$, and scale s, the output of f transforms coherently with the input, that is $f(\mathbf{TX}) = \mathbf{T}f(\mathbf{X})$. For the least trivial rotation component $\mathbf{R}$ of $\mathbf{T}$, [15] introduces a framework called Vector Neurons (VN) for constructing rotation-equivariant point cloud networks. By extending classical network neurons with ordered lists of scalars to ordered lists of 3D vectors $\mathbf{V} \in \mathbb{R}^{C\times 3}$ and introducing a set of equivariant layers, it allows the network activations to react to the rotations applied to the input data.

With the vector neuron building block, [6] then builds a $\mathrm{SIM}(3)$-equivariant shape encoding using implicit representations. Given a partial point cloud $\mathbf{X}$, a VN encoder Φ encodes the input into a latent code $\Theta = \Phi(\mathbf{X})$ comprised of four components $\Theta = (\Theta_R, \Theta_{\text{inv}}, \Theta_c, \Theta_s)$. Here, $\Theta_R \in \mathbb{R}^{C\times 3}$ is a rotation equivariant latent representation, $\Theta_{\text{inv}} \in \mathbb{R}^C$ is an invariant latent representation, scalar $\Theta_s \in \mathbb{R}$ represents object scale, and 3D vector $\Theta_c \in \mathbb{R}^3$ represents the object centroid. For any $\mathbf{T} = (\mathbf{R}, \mathbf{t}, s) \in \mathrm{SIM}(3)$, its action on Θ and the equivariance of encoder Φ can be written as

$$\mathbf{T}\Theta = (\Theta_R\mathbf{R}, \Theta_{\text{inv}}, s\Theta_c\mathbf{R} + \mathbf{t}, s\Theta_s) = \Phi(s\mathbf{XR} + \mathbf{t}). \quad (1)$$

More details of the encoder network can be found in [6]. The latent code can be decoded by an implicit decoder $\Psi(\mathbf{x}; \Theta)$ that takes a query position $\mathbf{x} \in \mathbb{R}^3$ as input and outputs a per-point feature $z(\mathbf{x})$ with

$$\Psi(\mathbf{x}; \Theta) = (\Theta_{\text{inv}}, \langle \Theta_R, (\mathbf{x} - \Theta_c)/\Theta_s \rangle). \quad (2)$$

Problem formulation. An agent is given a set of demonstrations $\mathcal{D}_{\text{demo}} = \{\tau_n\}_{n=1}^{N_{\text{demo}}}$. Each demonstration τ_i consists of a sequence of transition tuples $\{(o_t, a_t)\}_{t=1}^{T}$, where $o_t = (\mathbf{X}_t, \{\mathbf{y}_t^e\}_{e=1}^{E})$ is an observation that includes a point cloud of the scene $\mathbf{X}_t$ and the poses of all end-effectors $\{\mathbf{y}_t^e\}_{e=1}^{E}$, and a_t is an action that consists of velocity and gripper commands for all end-effectors. We train a policy $\pi(a|o)$ from the demonstrations to take an observation as input and output an action a. We then deploy the learned policy in a setup with an unseen object out of the distribution of objects in $\mathcal{D}_{\text{demo}}$ and is evaluated on the final reward of completing that target task.

B. Learning SIM(3)-equivariant 3D Visual Representations from Simulation Data

The first step of our method involves training a 3D representation for the objects involved in the manipulation task (illustrated in Fig. 2). To make the learned representation generalizable to diverse objects in the same category, we assume access to a dataset $\mathcal{D} = \{(\mathbf{X}^i \in \mathbb{R}^{N\times 3}, \mathbf{M}^i \in \mathbb{R}^{M\times 3})\}_{i=1}^{L_{\mathcal{D}}}$ containing point clouds of simulated scenes that include the objects of interest. In each sample, we record the partial point cloud of the scene unprojected from one camera view $\mathbf{X}^i$ and the ground-truth mesh of the objects in the scene $\mathbf{M}^i$. Since simulated scenes include full information about object meshes, we can assume that all ground-truth meshes $\mathbf{M}^i$ have the same number of vertices, and the points in two meshes correspond to each other – buying simplicity through this canonicalization.

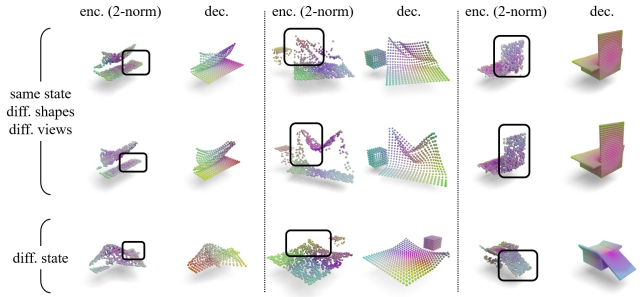

Fig. 3: **Visualizations of per-point features.** The encoder features are equivariant vector-valued features on the partial point cloud observations and the visualizations are done on their invariant components (channel-wise 2-norms). The decoder features are invariant scalar-valued features on the complete objects. The RGB values are computed via PCA within each task. All point clouds are aligned to the canonical pose for visualization. **Top two rows:** Objects of different shapes viewed from different camera angles but at the same poses. Both encoder and decoder features show strong correspondences within each state due to the contrastive learning. **Bottom row:** Objects from a different state. The features become different from the above rows.

We train the visual representation using a contrastive loss similar to [33]. More specifically, we sample a simulated data pair of object point clouds $(\mathbf{X}^1, \mathbf{M}^1)$ and $(\mathbf{X}^2, \mathbf{M}^2)$ where two different object instances share a similar pose, articulation, or deformation. Then, we aim to learn per-point features $\{z(\mathbf{m}_i^1)\}_{i=1}^{M}$ and $\{z(\mathbf{m}_i^2)\}_{i=1}^{M}$ at ground-truth mesh points. Since we know the two ground-truth meshes have one-to-one correspondences, we know that $z(\mathbf{m}_i^1)$ and $z(\mathbf{m}_i^2)$ correspond to the same points on the mesh and should have similar latent features. We then apply a PointInfoNCE loss [33] to train the latent representation:

$$\mathcal{L} = -\sum_{i\in[M]} \log \frac{\exp(z(\mathbf{m}_i^1) \cdot z(\mathbf{m}_i^2)/\tau)}{\sum_{j\neq i} \exp(z(\mathbf{m}_i^1) \cdot z(\mathbf{m}_j^2)/\tau)}. \quad (3)$$

Instead of the SDF decoder in [6] which is incompatible with thin-shell deformable objects with highly varying geometries (like cloth), or the contrastive loss on the multi-views of the same object as originally introduced in [33], we sample positive point pairs across different objects at comparable poses, which encourages representation to learn similar features for corresponding points of varied object instances.

C. Learning Generalizable Visuomotor Policies with SIM(3)-equivariant Architecture

After the representation learning phase, we freeze the encoder Φ and learn an action prediction head on top of the learned feature representation with a small number of human demonstrations. According to the explanations in [34], the intermediate representations from the encoder can possess

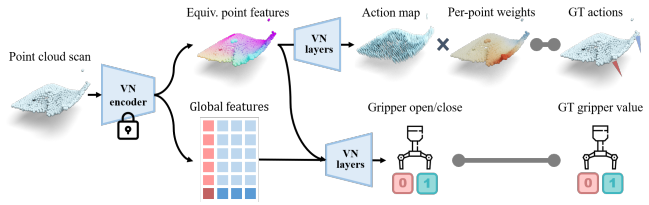

Fig. 4: **Policy learning architecture.** Given a point cloud scan during policy execution, we first pass it through a frozen encoder from the representation learning stage and acquire both local and global equivariant features. Two VN heads are then incorporated to predict the end-effector offsets and the gripper states.

desirable properties even without being decoded to the space where the contrastive loss is applied.

Architecture. The policy learning architecture is illustrated in Fig. 4. Given input observation $o = (\mathbf{X}, \{\mathbf{y}^e\}_{e=1}^E)$, our policy π first passes the input point cloud through the encoder Φ to obtain global feature Θ_R and per-point features $\{\mathbf{V}(\mathbf{x}_i)\}_{i=1}^N$ of the input scene point cloud. Then, the policy architecture is split into two branches. The offset prediction branch takes the local features as input and passes them through 3 VN layers to produce a velocity map $\{\hat{\mathbf{v}}_i\}_{i=1}^N$, which corresponds to the predicted velocity of the end-effector if the end-effector is close to the point location. The gripper action prediction branch takes the global feature as input and passes it through 3 VN layers different from the ones in the offset prediction branch to produce a binary gripper open/close prediction for each end-effector $\{\hat{g}_e\}_{e=1}^E$. During inference, we find for each end-effector $e \in \{1, \ldots, E\}$ a point $\mathbf{x}_{k_e}$ on the point cloud that is closest to the end-effector position $\mathbf{y}^e$ and then use the velocity map output of that position $\hat{\mathbf{v}}_{k_e}$ as the predicted velocity of the corresponding end-effector. The predicted end-effector velocity and gripper action predictions are then concatenated to form the output action $\hat{a} = (\{\hat{\mathbf{v}}_{k_e}\}_{e=1}^E, \{\hat{g}_e\}_{e=1}^E)$.

Loss function. We use a multi-part loss function that supervises the velocity predictions and the gripper control predictions of the policy separately. To supervise the outputs of the velocity map, we want to apply a loss term on every point on the point cloud. However, points that are located farther away from any end-effector in the demonstration should receive less supervision than points closer to one of the end-effectors. To achieve this, we weight the loss term applied to each point on the point cloud such that the closer a point is to the end-effector, the higher the loss is weighted. Concretely, the velocity prediction loss is defined by

$$\mathcal{L}_{\text{vel}} = \frac{1}{N} \sum_{i=1}^{N} \sum_{e=1}^{E} w(\mathbf{x}_i, \mathbf{y}_e) \cdot (\mathbf{v}_i - \hat{\mathbf{v}}_i),$$

where $\mathbf{v}_i$ denotes the ground truth velocity of the i-th end-effector, $\hat{\mathbf{v}}_i$ denotes the predicted velocity of the i-th end-effector, and $w(\mathbf{x}_i, \mathbf{y}_e) = \exp(-(\mathbf{x}_i - \mathbf{y}_e)^2/(2\sigma))$ is a weighting function that is large when $\mathbf{x}_i$ is close to $\mathbf{y}_e$ and small when $\mathbf{x}_i$ is far from $\mathbf{y}_e$. The gripper control prediction loss is simply defined as the mean squared error (MSE) loss between the predicted gripper actions $\hat{g}_e$ and the ground truth actions g_e: $\mathcal{L}_{\text{grip}} = \text{MSE}(\hat{g}_e, g_e)$. The policy learning loss is a weighted sum of the velocity prediction and gripper loss:

$$\mathcal{L} = \lambda_{\text{vel}} \mathcal{L}_{\text{vel}} + \mathcal{L}_{\text{grip}},$$

where λ_{vel} is a weighting term for the velocity part of the loss function.

IV. Experiments

Through our experiments, we wish to answer the following questions: **(Q1)** Can our method learn from a few demonstrations in one scenario and generalize to scenarios with unseen object sizes and poses? **(Q2)** Does our method outperform methods that rely on extensive augmentations to achieve generalization? **(Q3)** Does our method perform better than prior methods that do not employ equivariant architectures or pre-trained visual representations with paired simulation data?

A. Comparisons

We compare our method with several baselines and also analyze ablations, as described below.

(1) **PointNet+BC:** a baseline behavior cloning (BC) algorithm that trains a neural network architecture, which takes a partial point cloud of the scene as input, encodes the input with a PointNet [35], concatenates the point cloud feature with robot proprioception, and then passes the concatenated vector through an MLP to predict robot actions. Since this baseline does not have a representation learning phase, it only uses the source task demos and does not utilize simulated data. The purpose of this baseline is to examine a basic policy learning method one would use if one aims to train a policy that takes point cloud as input and outputs actions without worrying about generalization properties.

(2) **PointNet+BC$_{\text{aug}}$:** a variation of *PointNet+BC* that augments the demonstration data on object pose and scale, aiming to make test-time scenarios 'in distribution' with respect to the augmented data. The total amount of data used after augmentation equals the amount of data used in our method. This baseline serves to answer (Q2) from above.

(3) **PointContrast+BC:** this baseline first trains a latent representation of the 3D scene with PointContrast [33], and then learns an MLP that takes the learned features and robot proprioception as input to predict robot actions. This baseline serves to answer question (Q3).

(4) **PointContrast$_{\text{aug}}$+BC:** a variation of *PointContrast+BC* that uses augmented object poses during feature training, which is a way of approximating equivariant features with Siamese training [36].

(5) **Ours w/o pre-training:** this ablation is a variation of our method without a pre-training phase, which trains the whole architecture, including the visual encoder and the action prediction head, together with the policy.

(6) **Ours w/o equivariance:** this ablation uses a variation of our architecture that replaces all SIM(3) equivariant architecture with non-equivariant PointNet-based architecture with similar size.

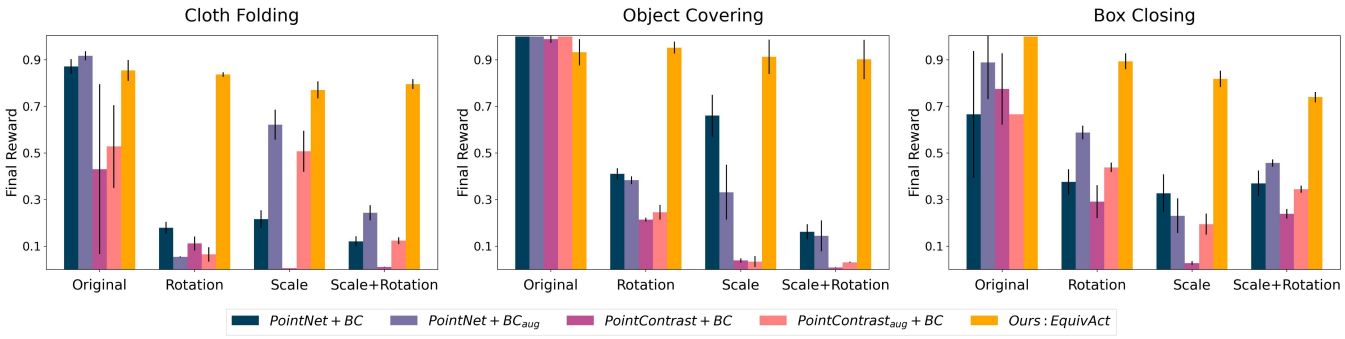

Fig. 5: **Results for simulation experiment.** We evaluate 3 manipulation tasks involving deformable and articulated objects. The comparisons with baselines show that our method outperforms prior methods that rely on augmentations to achieve generalization or utilize non-equivariant visual representations.

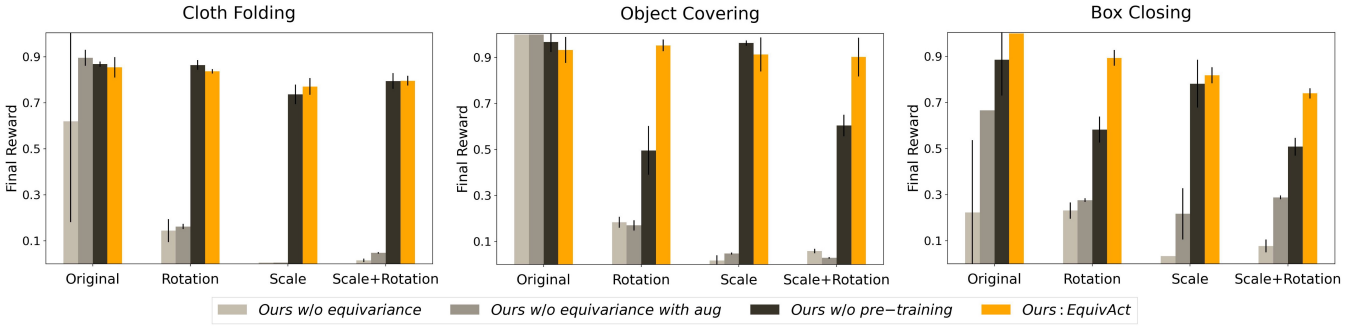

Fig. 6: **Ablation results in simulation.** The comparisons show that the pre-training and the equivariance components of our framework positively contribute to the performance of our framework.

(7) **Ours w/o equivariance with aug:** this ablation is our method without SIM(3) equivariance, but with using augmented object poses during feature training.

B. Simulation Experiments

Tasks. We evaluate our method on three challenging robot manipulation tasks involving various deformable and articulated objects. (1) *Cloth Folding:* two robots fold a piece of cloth together by grasping two corners of the cloth. This task is designed to test if our method can handle deformable object manipulation. (2) *Object Covering:* two robots grasp a cloth on two corners and place the cloth on top of an object so that the object is fully covered. This task is designed to test handling scenarios with several objects. (3) *Box Closing:* two robots close a box with three flaps by first closing the side flaps and then closing the larger front/back flap. This task tests manipulation with articulated objects.

Representation learning data. Our simulation environments use PyBullet simulation engine [37]. For simplicity, we use floating end-effectors, which can physically push objects and can also attach to objects to imitate grasping. We generate randomized simulation data in each task to be used for representation pre-training. In *Cloth Folding* and *Object Covering* tasks, we generate representation learning data by grasping two corners of the object and performing a scripted pick and place motion to random target positions around the cloth. We collected 200 episodes of simulation data in each of the two tasks, varying the non-uniform scaling and camera viewpoint of each episode. Note that these 200 episodes are not successful executions of the target task and do not include recorded robot actions. In the *Box Closing* task, we generate representation learning data by taking snapshots of the object at various articulated poses. We collect a total of 20k images, equivalent to 200 episodes of episode rollouts in this task. We vary the non-uniform scaling and camera viewpoint of each collected image.

Demonstration data. We collect 50 noisy demonstrations for all tasks. In each task, the 50 demos include the same object at the canonical pose. This means that the policy is trained using demonstrations of only one manipulation scenario.

Evaluation. To thoroughly examine the generalization capability of the learned policy, we test our policy in four different setups. The *Original* setup evaluates the performance of the policy with an object that is placed and sized the same as in the demonstrated scenario; the *Rot* setup randomizes the rotation of the object while keeping the object scale constant; the *Scale* setup randomizes the non-uniform scale of the object between $1\times$ and $2\times$ while keeping the change of aspect ratio within $1 : 1.33$ (shortest vs. longest dimensions) and keeping the object in a canonical pose; while the *Rot+Scale* setup randomizes both the rotation and non-uniform scaling of the object. Note that except for the *Original* setup, all other setups test the out-of-distribution performance of the trained policy. In particular, in the *Box Closing* task, we add a further challenge to the trained policy by removing the contents in the box at test time, so that the test-time objects have different shapes compared to the training-time object.

We measure the performance of a task using a task-specific

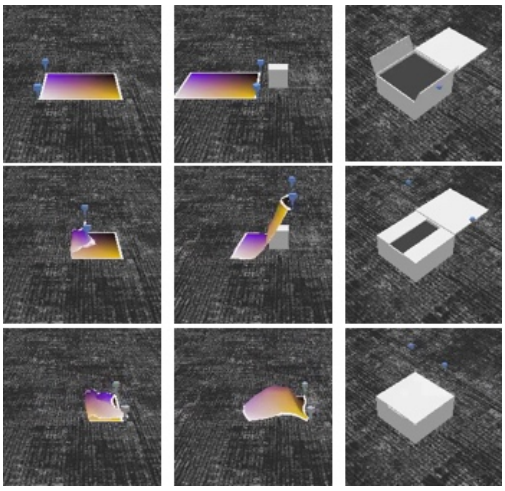

Fig. 7: **Demonstrations for simulated experiments.** From left to right, the figure shows the cloth folding, object covering, and box closing tasks.

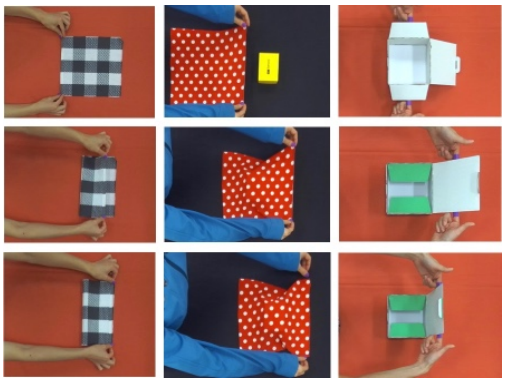

Fig. 8: **Human demonstrations collected for real robot experiments.** From left to right, the figure shows the cloth folding, object covering, and box closing tasks.

reward function scaled between 0 and 1. In the *Cloth Folding* task, the reward is measured by how close the bottom two corners of the cloth are to the top two corners of the cloth; in the *Object Covering* task, the reward equals the percentage of the object volume within the convex hull of the final cloth mesh; in the *Box Closing* task, the reward equals the average percentage that each of the three flaps of the box is closed.

Results. Fig. 5 shows experimental results in simulation. In the *Original* evaluation setup, all methods exhibit good performance. This shows that all the methods have learned to complete the manipulation tasks that were demonstrated. In the other three setups, the two **PointNet+BC** baselines display a significant performance drop. Even after adding augmentations in **PointNet+BC$_{\text{aug}}$**, the baseline is still not able to recover the in-distribution performance for any of the three tasks. This shows that augmentation itself does not guarantee effective generalization towards varied object scales and object poses. Similarly, the **PointContrast+BC** baseline performs well in the *Rot* setup but struggles to achieve good performance in the evaluation setups that involve scaling. This shows that our pre-training strategy that utilizes paired data with objects in different non-uniform scales helps our method cope with randomized non-uniform scaling of the objects in the *Scale* and *Rot+Scale* evaluation

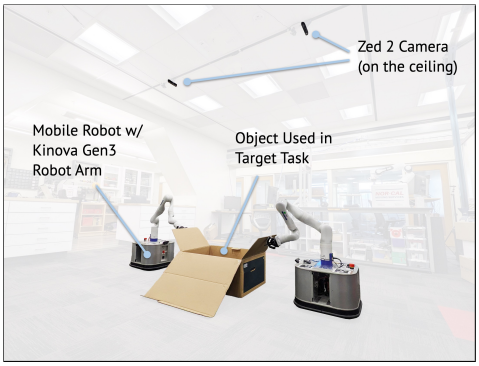

Fig. 9: **Real robot setup.** Our setup consists of two Zed cameras on the ceiling used for getting point clouds of the scene, two mobile bases with Kinova Gen3 arms mounted on top, and a target object to be manipulated.

Fig. 10: **Qualitative samples of deploying the policy on hardware.** We deploy our learned policies to a mobile robot platform to manipulate target objects that are at least 6 times larger than the demonstrated objects. They have different visual appearances, aspect ratios, and physical properties (e.g. friction and elasticity of the cloth) than those in the source tasks. Even with such a large difference between source and target domains, our policy can zero-shot generalize to the target objects and can successfully carry out the task. Please see our website for full videos of these samples.

setups. Our method achieves good performance in all out-of-distribution setups, displaying only a minor performance drop compared to the simpler *Original* setup.

Our ablation experiments are presented in Fig. 6. The performance of the **Ours w/o Pretraining** and **Ours w/o Equivariance** ablations show that having pre-training and equivariance in our framework positively contribute to the performance of our framework.

C. Real Robot Experiments

To illustrate the effectiveness of our method in the real world, we test it with both quantitative and qualitative experiments in a mobile manipulation setup.

Tasks and data collection. We run our method in the same three manipulation tasks as our simulation experiments. In each task, our method takes 20 human demonstrations collected on a tabletop setup as training data. In the *Cloth Folding* task, we use a square cloth with a side length of 27.5cm; in the *Object Covering* task, we use a square cloth with a side length of 27.5cm to cover a box of size $13 \times 8 \times 8$cm; in the *Box Closing* task, we use a small $15 \times 13.5 \times 9.5$cm electronics box with three flaps. We

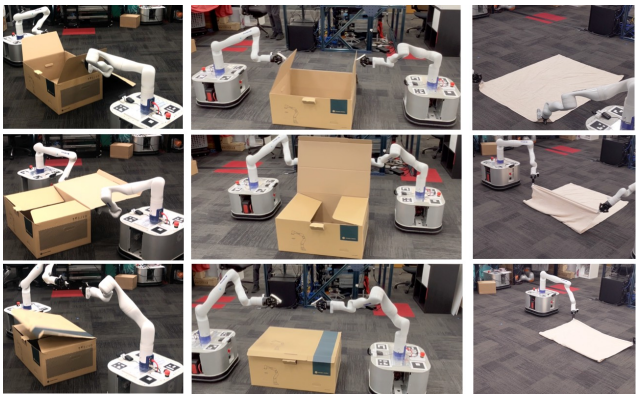

Fig. 11: **Task variations.** To illustrate the generalization capability of our method, we test it on various objects and initial poses. The first two columns show *Box Closing* policy on two differently sized boxes placed in different initial rotations. The same policy successfully closes the boxes in both cases. The third column shows that the *Cloth Folding* policy can handle different cloths with distinct appearance, scale, and physical properties compared to the pink cloth blanket in Fig. 10. Our website provides further examples.

use a ZED 2 stereo camera positioned above the table to record the movement of the objects and human hands. After data collection, we segment out the part of the point cloud that corresponds to the objects the human is manipulating and also parse out the human finger positions in each frame. We treat segmentation as a separate research problem, which could be addressed by state-of-the-art methods, such as [38]. However, to avoid conflating the evaluation of the performance of various segmentation methods with the main evaluation of our method, we segment out the objects using simple (robust) color filtering techniques.

Mobile robot setup. We train our method on data extracted from human demonstrations and then deploy it in the real world. Our real robot setup is illustrated in Fig. 9. The robots operate in a large workspace with a size 4×3 meters. We use holonomic mobile bases with a powered-caster drive system [39], and Kinova Gen3 7-DoF arms equipped with Robotiq 2F-85 grippers. We use Zed 2 stereo cameras positioned on the ceiling to obtain point clouds of the scene, then segment out the relevant parts of the scene. As before, we treat object segmentation as a problem beyond the scope of this work and use simple color filtering to separate the objects from the background and robots. The policy takes the partial point cloud of the objects in the scene as well as the proprioception readings of the two mobile robots as input, then outputs a velocity and gripper open/close command for each robot. For simplicity, we initialize the robot end-effectors close to the object and select an appropriate end-effector rotation for completing the task.

Evaluation. We compare our method with **PointNet+BC$_{\text{aug}}$**, the best-performing baseline in the simulation experiments. For each method, we run 10 trials with the learned policy. Between each trial, we randomize the position of the objects in a $\pm 0.2m$ range and the orientation of the objects in a $\pm 15^{\circ}$ range.

To evaluate the performance of the learned policies in the real world, we design simple evaluation metrics for each of the three tasks. In the *Cloth Folding* task, we measure performance by computing the average distance between the grasped and target cloth corners; in the *Object Covering* task, we evaluate performance by measuring roughly how much of the top of the box ends up being covered by the cloth (eg. 0%, 25%, 50%, 75%, 100%); in the *Box Closing* task, we evaluate performance by computing the percentage of box flaps the agent completely closes during the episode.

	Cloth Folding	Object Covering	Box Closing
PointNet+BC$_{\text{aug}}$	0.083	0.000	0.000
Ours (EquivAct)	**0.919**	**0.825**	**0.867**

TABLE I: **Quantitative results in real robot experiments.** In a real-world mobile manipulation platform, we compare our method against the baseline that performed best in simulation experiments. We use simple evaluation metrics scaled in a range of 0 to 1 for all tasks. In the table, we show the average performance of running each learned policy for 10 trials on the real robot platform while randomizing the positions and orientations of the objects of interest. The result shows that in all three tasks, our method is able to successfully complete the target manipulation task, while the baseline cannot reliably finish any task.

Quantitative results. We present our quantitative results in Table I. In all three tasks, our method extrapolates from tabletop-scale demonstrations and successfully manipulates objects that are up to $6\times$ larger in size. The **PointNet+BC$_{\text{aug}}$** baseline, on the other hand, has a lot of trouble when attempting all three tasks. The common failure mode of the baseline is that it keeps predicting robot actions in one direction. This is likely because the visual input received by the baseline is out-of-distribution, and the method is not able to generalize to these unseen cases.

Qualitative results. We further demonstrate the robustness of our method by varying the position, rotation, scale, and appearance of the objects in the task (see Fig. 11). Our qualitative evaluations show that our method can robustly generalize to all these variations of the task.

V. Conclusions and Future Work

We presented *EquivAct*, a visuomotor policy learning method that learns generalizable closed-loop policies for 3D manipulation tasks. We showed that our method successfully generalizes from a small set of demonstrations to a diverse set of target scenarios in both simulation and real robot experiments.

While our work presented promising results, several limitations are worth addressing in future work. Our method uses a simple behavior cloning algorithm that is not robust against imperfect demonstrations or multi-modal behaviors. The overall performance could be improved with a more advanced imitation learning approach, e.g. using transformer-based architectures. We also note that we currently regard segmentation of objects in the scene as a separate research problem. To build a more streamlined system, one could employ advanced segmentation methods for scene parsing.

Overall, we hope that our work opens up opportunities for follow-up works and systems that utilize the benefits of equivariant architectures and facilitates learning generalizable robot manipulation policies.

REFERENCES

[1] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis, "Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 627–12 637.

[2] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull, "Active domain randomization," in *Conference on Robot Learning*. PMLR, 2020, pp. 1162–1176.

[3] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, "Reinforcement learning with augmented data," *Advances in neural information processing systems*, vol. 33, pp. 19 884–19 895, 2020.

[4] N. Hansen and X. Wang, "Generalization in reinforcement learning by soft data augmentation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 13 611–13 617.

[5] T. Yu, T. Xiao, A. Stone, J. Tompson, A. Brohan, S. Wang, J. Singh, C. Tan, J. Peralta, B. Ichter, *et al.*, "Scaling robot learning with semantically imagined experience," *arXiv preprint arXiv:2302.11550*, 2023.

[6] J. Lei, C. Deng, K. Schmeckpeper, L. Guibas, and K. 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*, 2023, pp. 4902–4912.

[7] A. Simeonov, Y. Du, A. Tagliasacchi, J. B. Tenenbaum, A. Rodriguez, P. Agrawal, and V. Sitzmann, "Neural descriptor fields: Se (3)-equivariant object representations for manipulation," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 6394–6400.

[8] A. Simeonov, Y. Du, Y.-C. Lin, A. R. Garcia, L. P. Kaelbling, T. Lozano-Pérez, and P. Agrawal, "Se (3)-equivariant relational rearrangement with neural descriptor fields," in *Conference on Robot Learning*. PMLR, 2023, pp. 835–846.

[9] Z. Xue, Z. Yuan, J. Wang, X. Wang, Y. Gao, and H. Xu, "Useek: Unsupervised se (3)-equivariant 3d keypoints for generalizable manipulation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 1715–1722.

[10] H. Ryu, H.-i. Lee, J.-H. Lee, and J. Choi, "Equivariant descriptor fields: Se (3)-equivariant energy-based models for end-to-end visual robotic manipulation learning," *arXiv preprint arXiv:2206.08321*, 2022.

[11] T. Weng, D. Held, F. Meier, and M. Mukadam, "Neural grasp distance fields for robot manipulation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 1814–1821.

[12] N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff, and P. Riley, "Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds," *arXiv preprint arXiv:1802.08219*, 2018.

[13] F. Fuchs, D. Worrall, V. Fischer, and M. Welling, "Se (3)-transformers: 3d roto-translation equivariant attention networks," *Advances in neural information processing systems*, vol. 33, pp. 1970–1981, 2020.

[14] H. Chen, S. Liu, W. Chen, H. Li, and R. Hill, "Equivariant point network for 3d point cloud analysis," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 14 514–14 523.

[15] C. Deng, O. Litany, Y. Duan, A. Poulenard, A. Tagliasacchi, and L. J. Guibas, "Vector neurons: A general framework for so (3)-equivariant networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 200–12 209.

[16] S. Assaad, C. Downey, R. Al-Rfou, N. Nayakanti, and B. Sapp, "Vn-transformer: Rotation-equivariant attention for vector neurons," *arXiv preprint arXiv:2206.04176*, 2022.

[17] O. Katzir, D. Lischinski, and D. Cohen-Or, "Shape-pose disentanglement using se (3)-equivariant vector neurons," in *European Conference on Computer Vision*. Springer, 2022, pp. 468–484.

[18] J. Li, S. Luo, C. Deng, C. Cheng, J. Guan, L. Guibas, J. Ma, and J. Peng, "Orientation-aware graph neural networks for protein structure representation learning," 2022.

[19] A. Poulenard and L. J. Guibas, "A functional approach to rotation equivariant non-linearities for tensor field networks." in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 13 174–13 183.

[20] C. Deng, J. Lei, B. Shen, K. Daniilidis, and L. Guibas, "Banana: Banach fixed-point network for pointcloud segmentation with inter-part equivariance," *arXiv preprint arXiv:2305.16314*, 2023.

[21] H.-X. Yu, J. Wu, and L. Yi, "Rotationally equivariant 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 1456–1464.

[22] X. Liu, J. Zhang, R. Hu, H. Huang, H. Wang, and L. Yi, "Self-supervised category-level articulated object pose estimation with part-level se (3) equivariance," *arXiv preprint arXiv:2302.14268*, 2023.

[23] R. Kondor and S. Trivedi, "On the generalization of equivariance and convolution in neural networks to the action of compact groups," in *International Conference on Machine Learning*. PMLR, 2018, pp. 2747–2755.

[24] T. S. Cohen, M. Geiger, and M. Weiler, "A general theory of equivariant cnns on homogeneous spaces," *Advances in neural information processing systems*, vol. 32, 2019.

[25] M. Weiler, P. Forré, E. Verlinde, and M. Welling, "Coordinate independent convolutional networks–isometry and gauge equivariant convolutions on riemannian manifolds," *arXiv preprint arXiv:2106.06020*, 2021.

[26] J. Aronsson, "Homogeneous vector bundles and g-equivariant convolutional neural networks," *Sampling Theory, Signal Processing, and Data Analysis*, vol. 20, no. 2, p. 10, 2022.

[27] Y. Xu, J. Lei, E. Dobriban, and K. Daniilidis, "Unified fourier-based kernel and nonlinearity design for equivariant networks on homogeneous spaces," in *International Conference on Machine Learning*. PMLR, 2022, pp. 24 596–24 614.

[28] M. Jia, D. Wang, G. Su, D. Klee, X. Zhu, R. Walters, and R. Platt, "Seil: Simulation-augmented equivariant imitation learning," *ICRA*, 2023.

[29] H. Ha and S. Song, "Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding," in *Conference on Robot Learning*. PMLR, 2022, pp. 24–33.

[30] C. Chi and S. Song, "Garmentnets: Category-level pose estimation for garments via canonical space shape completion," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 3324–3333.

[31] T. Weng, S. M. Bajracharya, Y. Wang, K. Agrawal, and D. Held, "Fabricflownet: Bimanual cloth manipulation with a flow-based policy," in *Conference on Robot Learning*. PMLR, 2022, pp. 192–202.

[32] B. Shen, Z. Jiang, C. Choy, L. J. Guibas, S. Savarese, A. Anandkumar, and Y. Zhu, "Acid: Action-conditional implicit visual dynamics for deformable object manipulation," *arXiv preprint arXiv:2203.06856*, 2022.

[33] S. Xie, J. Gu, D. Guo, C. R. Qi, L. Guibas, and O. Litany, "Pointcontrast: Unsupervised pre-training for 3d point cloud understanding," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*. Springer, 2020, pp. 574–591.

[34] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[35] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[36] W. Sun, A. Tagliasacchi, B. Deng, S. Sabour, S. Yazdani, G. E. Hinton, and K. M. Yi, "Canonical capsules: Self-supervised capsules in canonical pose," *Advances in Neural information processing systems*, vol. 34, pp. 24 993–25 005, 2021.

[37] E. Coumans and Y. Bai, "PyBullet, a Python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2019.

[38] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, *et al.*, "Segment anything," *arXiv preprint arXiv:2304.02643*, 2023.

[39] R. Holmberg and O. Khatib, "Development and control of a holonomic mobile robot for mobile manipulation tasks," *The International Journal of Robotics Research*, vol. 19, no. 11, pp. 1066–1074, 2000.